Routing Protocols for Next Generation Networks Inspired by Collective Behaviors of Insect Societies: An Overview *

Muddassar Farooq[1] and Gianni A. Di Caro[2]

[1] Next Generation Intelligent Networks Research Center
National University of Computer and Emerging Sciences (NUCES)
Islamabad, Pakistan
muddassar.farooq@udo.edu

[2] "Dalle Molle" Institute for Artificial Intelligence (IDSIA)
Lugano, Switzerland
gianni@idsia.ch

Summary. In this chapter we discuss the properties and review the main instances of network routing algorithms whose bottom-up design has been inspired by *collective behaviors of social insects* such as *ants* and *bees*. This class of bio-inspired routing algorithms includes a relatively large number of algorithms mostly developed during the last 10 years and mainly inspired by ant colony behaviors. It accounts for the majority of the instances of swarm intelligence algorithms for routing. The characteristics inherited by the biological systems of inspiration almost naturally empower these algorithms with characteristics such as *autonomy, self-organization, adaptivity, robustness*, and *scalability*, which are all desirable if not necessary properties to deal with the challenges of current and next generation networks. In the chapter we consider different classes of wired and wireless networks, and for each class we briefly discuss the characteristics of the main ant- and bee-colony inspired algorithms which can be found in literature. We point out their their distinctive features and discuss their general pros and cons in relationship to the state-of-the-art.

1 Introduction

The constant improvement in communication technologies and the related dramatic increase in user demand to be connected anytime and anywhere to both the wealth of information accessible through the Internet and other users and communities, have boosted the pervasive deployment of wireless and wired networked systems. These systems are characterized by the fact of being *large*

* Draft version of the chapter appeared in the book: Blum C., Merkle D. (Eds.), *Swarm Intelligence: Introduction and Applications*, Springer, Natural Computing Series, 2008.

or very large, highly *heterogeneous* in terms of communication technologies, protocols, and services, and very *dynamic*, due to continual changes in topology, traffic patterns, and number of active users and services. *Intelligent* [10] and *autonomic* [73] management, control, and service provisioning in these complex networks, and in the future networks resulting from their integration and evolution, require the definition of novel protocols and techniques for all the architectural components of the network.

In this chapter we focus on the *routing* component, which is at the very core of the functioning of every network since it implements the strategies used by network nodes to discover and use paths to forward data/information from sources to destinations. An effective design of the routing protocol can provide the basic support to unleash the intrinsic power of the highly pervasive, heterogeneous, and dynamic, complex networks of the next generation. In this perspective, the routing path selection must be realized in a fully *automatic* and *distributed* way, and it must be *dynamic*, to take into account the constant evolution of the network state, which is defined by multiple concurrent factors such as topology, traffic flows, available services, etc.

The literature in the domain of routing is very extensive. Routing research has fully accompanied the evolution of networking to constantly adapt the routing protocols to the different novel communication technologies and to the changes in user demand. In this chapter we review routing protocols and algorithms which have been specifically designed taking inspiration from, and reverse engineering the characteristics of, processes observed in *insect societies*. This class of routing protocols is indeed relatively large. The first notable examples date back to the beginning of the second mid of the 90's [113, 27, 123, 151], and a number of further implementations were rapidly following the first ones and gained the attention of the scientific community. In the following of the chapter we will limit the discussion to the most popular and effective instances of this specific class of routing protocols.

The fact that insect societies, and, more in general, nature, has served as a major source of inspiration for the design of novel routing algorithms can be understood by noticing that these biological systems are characterized by the presence of a set of distributed, autonomous, minimalist units, that through local interactions self-organize to produce system-level behaviors which show life-long adaptivity to changes and perturbations in the external environment. Moreover, these systems are usually resilient to minor internal failures and losses of units, and scale quite well by virtue of their modular and fully distributed design. All these characteristics, both in terms of system organization and resulting properties, meet most of the necessary and desired properties of routing protocols for next generation networks. This fact makes potentially very attractive looking at insect societies to draw inspiration for the design of novel routing protocols featuring *autonomy, distributedness, adaptivity, robustness*, and *scalability*. These are desirable properties not only in the domain of network routing but also in a number of other domains. As a matter of fact, in the last 20 years, collective behaviors of insect societies

related to operations such as *foraging, labor division, nest building and maintenance, cemetery formation*, etc., have provided the impetus for a growing body of scientific work, mostly in the fields of telecommunications, distributed systems, operations research, and robotics (e.g., see [48, 24, 7, 43, 46] for references and overviews). Behaviors observed in colonies of *ants* and of *termites* have fueled the large majority of this work. More recently, also *bee colonies* are attracting a growing interest. In the following we precisely review network routing algorithms inspired by these three classes of social insects. The vast majority of the reviewed algorithms are derived from ant colonies, and in particular, from their ability to discover and follow *shortest paths* between their nest and sources of food [60].

All the algorithms that we will discuss later in the chapter are characterized by the fact of being composed by a potentially very large number of *autonomous* and *fully distributed* controllers, and of having been designed according to a *bottom-up* approach relying on basic *self-organizing* abilities of the system. These characteristics, together with the biological inspiration from behaviors of *insect societies*, are the very fingerprints of the *swarm intelligence* (SI) paradigm [7]. These peculiar design guidelines contrast with those of the more common *top-down* approach followed for the design of the majority of "classical" routing protocols. In typical top-down design a centralized algorithm with well-known properties is implemented in a distributed system. Clearly, this requires to modify the original algorithm to cope with the intrinsic limitations of a distributed architecture in terms of full state observability and delays in the propagation of the information. The main effect of these modifications consists in the fact that several properties of the original algorithm do not hold anymore if the network dynamics is non-stationary, which is the most common case. Still, it is relatively easy to assert some general formal properties of the system. On the other hand, with the bottom-up approach, the design starts with the definition of the behavior and interaction modalities of the individual node in the perspective of obtaining the wanted global behavior as the result of the joint actions of all nodes interacting with one another and with the environment at the local level. It is in general "easier" to follow a bottom-up approach, and the resulting algorithm is usually more flexible, scalable, and capable to adapt to a variety of different situations. This is precisely the case for the SI algorithms that we will review. The negative aspect of this way of proceeding is that is usually hard to state the formal properties and the expected behavior of the system. One of the objectives of this chapter consists in showing the common traits and properties of SI routing algorithms derived from insect societies, and compare them to the characteristics and properties of established state-of-the-art routing algorithms not based on SI, and evaluate the relative merits.

For space reasons and without loss of generality, we will restrict the classes of networks that we will consider. More specifically, we will focus the discussion on routing algorithms for non-optical *connectionless* and *connection-oriented wired networks* offering *best-effort* and/or *guaranteed quality* services, and for

wireless mobile ad hoc networks (MANETs) [106]. These are wide and general classes of networks that include a large number of network instances of both practical and theoretical interest. Concerning SI-based routing algorithms for other important classes of networks the interested reader can consult for instance [91, 59] for the case of *optical networks*, [119] for the case of *satellite networks*, and [88, 97, 15, 108, 109] for *sensor networks*. In [145] the interested reader can find a general overview of nature-inspired routing algorithms, while in [2], she/he can find a more general discussion on the design of algorithms for modern telecommunication networks using design patterns derived from the observation of biological systems.

1.1 Organization of the chapter

The remaining content of the chapter is organized considering separately the ant and the bee colony inspired frameworks and their applications to each one of the considered classes of telecommunication networks. For each algorithm we will point out the general design characteristics and performance.

- Section 2 briefly introduces *network routing*, and discusses the general characteristics of routing and the associated challenges for each one of the considered network classes.
- Section 3 provides a comprehensive set of *classification features* that we will use to characterize routing protocols and to which we will refer to throughout the chapter to highlight the main differences among the different protocols and, more specifically, between the SI protocols reviewed here and the more standard, established ones which are widely deployed in real-world networks.
- Section 4 and its two subsections describe respectively the ant and bee colony behaviors that have fueled the design of so many network routing algorithms. In particular, Subsection 4.1 introduces the *Ant Colony Optimization (ACO)* metaheuristic, which is based on the reverse-engineering of the ant colony shortest path behavior, and which has provided the main practical guidelines for the design of the ant colony inspired algorithms.
- Section 5 and all its subsections are devoted to the discussion of routing protocols derived from ACO. First, the general principles behind ACO and ACO for routing are discussed in Subsection 5.1. In Subsections 5.2 and 5.3, we describe in some detail AntNet and ABC, which are the main reference algorithms that have guided the design of most of the other algorithms. In Subsections 5.4 to 5.7 we discuss the characteristics of a number of ACO routing algorithms. The algorithms are grouped per network type and are considered in chronological order.
- Section 6 and its two subsections are devoted to the discussion of routing protocols derived from bee colonies. In practice, we discuss in some detail two main implementations, BeeHive for wired connectionless networks, and BeeAdHoc for MANETs.

- Section 7 summarizes the presented results and draws some general conclusions about the efficacy and the future perspectives of the SI approach to the design of novel routing protocols for next generation networks.

2 Generalities and Challenges of Network Routing

The behavior of the network routing protocol drives network dynamics and critically affects performance. In fact, it implements the strategies used by network nodes to determine and use paths to forward data/information from sources to destinations. Generally speaking, the *routing protocol* defines what information is going to be used to take routing decisions, how this information is communicated among the nodes, and how it is encoded in the node *routing table*, which is the local database of routing information. A routing table maintains the necessary information to define for each end node of interest and for each one of the locally available output interfaces the quality/cost associated to the selection of the specific interface as the next hop to forward data toward the end node. The *routing algorithm* part of the protocol makes use of this information to actually select the paths and forward data along them. The challenges faced by a routing protocol and the measure of its efficacy depend on the characteristics of the network at hand. In the following of this section we briefly review these aspects for the considered network types. The interested reader can find more accurate discussions in networking textbooks.

Transmission mode: Connection-oriented vs. connectionless

One basic distinction among network types is based on the adopted point-to-point *switching* technique. The two main classes of networks can be singled out: *circuit-switched* and *store-and-forward*. In circuit-switched networks, prior to start sending end-to-end data, it is necessary to seek out and establish a physical dedicated path between the two end points. No buffers for data are needed. Once the connection is setup, the only delay is propagation time. A telephone network is a typical example of a circuit-switched network. In store-and-forward networks, an intermediate node along the path stores each incoming block of data, inspect it for errors, and retransmit it along the path to the destination. *Message*, *packet*, and *cell switching*, refer respectively to the cases of a store-and-forward network in which the transferred block is a complete message, a variable-length block of data with a size upper bound, or a small, fixed-size block of data. The most widely in use switching method in networks, such as the Internet, is the *packet-switching* one. It can support different transmission modes. The *connection-oriented* mode shares the same principles as the circuit-switching technique. Prior to packet sending, a path connection (*virtual circuit*) must be established between the two end-points. The virtual circuit can be a dedicated physical connection or a logical one, shared among different data sessions. The task of the routing system is to find

and use full end-to-end paths. Typical measures of performance in this case are the *session acceptance ratio*, the *delivered throughput*, and statistics of the packet latencies such as the *average end-to-end delay*. The latter two performance metrics are reference metrics for almost any type of network, since they summarize two basic aspects related to the quantity and the quality of the service a network can deliver. In connectionless (*datagram*) networks, a packet is injected into the network without requiring establishing any connection, physical or virtual, and without any guarantee that the packet will be delivered at the destination. Each relay node deals with the packet independently from the other nodes and makes use of packet header information to decide how to route the packet. In this case, routing tables and data forwarding across the nodes should be consistent to let the packet traveling over existing and loop-less routes

Delivered service: Best-effort vs. guaranteed-quality

One major distinction can be done between networks offering *best-effort* services and those offering *quality-of-service (QoS)*. In best-effort networks the user applications are served with no guarantees on the quality of the delivered service. On the other hand, in QoS networks the user can specify constraints on the quality of the obtained service (e.g., in terms of end-to-end delay, delay jitter, bandwidth, etc.) and the network is expected to either meet these requirements or reject the application. In QoS networks the general challenge of routing consists in the ability to rapidly and robustly identify one or more paths that meet the QoS requirements of current traffic sessions while providing at the same time an efficient utilization of the network resources in order to be ready to satisfy the QoS requests of also future sessions. There are several network models that can allow provisioning of QoS. The most popular ones are: *IntServ, DiffServ*, and *Multi Protocol Label Switching (MPLS)* (e.g., see [139]). In IntServ the network must find and reserve resources for each single QoS flow. DiffServ is based on the organization of data traffic in multiple classes, with each class associated to different QoS requirements. Each packet is placed into a specific class and each router is configured to take different routing and scheduling actions depending on the class of the data packet. MPLS is a data-carrying mechanism which emulates some basic properties of a circuit-switched network over a packet-switched network. Once an end-to-end path has been found, it is uniquely identified at the nodes by means of labels and can be then efficiently used to forward data flows.

Topology and connectivity: Wired vs. wireless mobile ad hoc networks

In *wired networks* hosts and routers are connected through one-to-one cables creating a fixed network topology which undergoes only low-rate modifications due to addition/removal of resources and to temporary failures. Point-to-point

communication links are usually reliable and have large bandwidth. Terminals are equipped with good computational resources and are not concerned by power supply issues. The challenges for a routing protocol are the changing traffic patterns, the heavy loads, the small topological modifications, and the usually large number of nodes which scale up over time.

Wireless networks with mobile users present radically different characteristics and challenges. In this chapter we are interested in one specific class of wireless mobile networks, the *mobile ad hoc networks (MANETs)* [106], which during the past few years have become a very active area of research due to their unique characteristics. In a MANET all nodes are mobile and can enter and leave the network at any time. They communicate with each other via medium-range wireless connections that can constantly be established and broken because of mobility. There is no ground infrastructure to rely on. All nodes are peers and can serve as routers to each other. Data packets are forwarded from node to node in a multi-hop fashion. The wireless channel is shared among the peer nodes and the access must be arbitrated according to some distributed Medium Access Control (MAC) protocol, which results in a rather low and irregular amount of effective available bandwidth. Terminals have usually less computational power than in the wired case and are powered by on-board batteries with limited lifetime. All these aspects such as mobility, shared channel, low bandwidth, short battery lifetime, and distributed multi-hop forwarding, impose severe challenges and restrictions to the routing protocol. A good protocol is one that can effectively *adapt* to dramatic topological changes, needs relatively *low control overhead*, provides *high throughput* and *low packet delays*, and saves as much as possible of *battery power* to let the users and their mobile devices participate as long as possible to the network activities. It is clearly very hard to meet in a satisfactory way all these conflicting objectives, therefore, a rather large number of different routing algorithms have rapidly appeared in literature (e.g., see [106, 11, 122]). A common feature of MANET routing algorithms is that they are all adaptive.

State-of-the-art routing algorithms

Long-standing research on network routing has resulted in a rather large number of routing protocols and algorithms showing different characteristics according to the different types of networks and offered services they are meant for. Clearly, it is not possible to properly account for this large literature here. In this paragraph we limit ourselves to a brief discussion of a small number of state-of-the-art algorithms that are often mentioned to assess the relative performance of the reviewed swarm intelligence algorithms.

`OSPF` [87] and `RIP` [80] are among the most popular protocols for routing within Autonomous Systems (*interior protocols*) in use in the wired Internet, while BGP [158] is widely used to communicate among Autonomous Systems. OSPF belongs to the category of *link-state* algorithms. In these algorithms, each node periodically floods a comprehensive state description of

all its communication links. This description is used at each receiving node to incrementally construct and update a complete weighted graph of the network. OSPF makes use of Dijkstra's shortest path algorithm to calculate the routes based on this graph representation. While OSPF is mainly topology-adaptive, an earlier version of it, `Shortest Path First (SPF)` [74], was both topology- and traffic-adaptive. `QOSPF` [159] is an extension of OSPF to deal with QoS requests in conjunction with a resource reservation protocol such as *RSVP* [157]. In QOSPF, flooded link state messages report about QoS information and resources used by active flows.

RIP and BGP are instances of *distance-vector* protocols. In this case, each node only knows the set of network destinations and maintains in the routing table the vector of the best distances (e.g., number of hops) to reach each destination. These distances are periodically sent to all the neighbors and are calculated incrementally from hop to hop using algorithms derived from the well-known `Distributed Bellman-Ford` algorithm [6], which is in turn based on *dynamic programming* [5]. In practice, when node i receives from its neighbor j a message saying that j's shortest distance estimate to destination d is of n hops, i can safely set its best distance to d as $n+1$ hops if its current shortest distance estimate was $m > n+1$. This way of constructing distance estimates is prone to what is termed "counting to infinity": a very slow convergence to the right distance vectors after a destination becomes unreachable, with the concrete risk of incurring in loops and dangling routes. A notable recent distance-vector implementation which deals effectively with these problems and has also interesting additional properties is the `Multi-path Distance Vector Algorithm (MDVA)` [138]. The algorithm is loop-free under stationary conditions and makes also use of multiple-paths.

The Bellman-Ford's way of constructing estimates building on others' estimates is also termed *bootstrapping* and is widely used in the domain of *model-based reinforcement learning* [125]. More precisely, the notions of bootstrapping and reinforcement learning have guided the design of `Q-routing` [9] and of the derived `PQ-routing` [19], which are among the most notable contributions of artificial intelligence research to the domain of network routing.

Concerning MANETs, the reference algorithms are: `Ad-hoc On-demand Distance Vector routing (AODV)` [99], `Optimized Link State Routing (OLSR)` [21], and `Dynamic Source Routing (DSR)` [70]. AODV is a *reactive distance-vector* algorithm, that is, routing information is only collected when necessary to route an active traffic session. OLSR is a *proactive link-state* algorithm directly derived from OSPF and adapted to deal with the dynamic aspects of MANETs. DSR is a *reactive source routing* algorithm, that is, the header of each data packet carries the complete route to the destination in the form of ordered next hop nodes.

3 Classification Features of Network Routing Protocols

In principle, many different taxonomies can be adopted to effectively classify routing protocols (e.g., [54]). In the following, we identify a specific set of classification features which will serve to capture the distinctive characteristics of each considered SI algorithms and, at the same time, to point out the general differences existing between these algorithms and more standard, non nature-inspired, protocols. The classification features we propose here are partly based on those considered by CISCO [20]:

Static vs. Dynamic. Static routing protocols are based on the use of routing tables which are defined *offline* by network administrators according to some prior knowledge on the network. Dynamic protocols update routing tables and routing decisions *online* to reflect changes in the network state. Most of the protocols currently in use on the Internet, such as the mentioned OSPF and RIP mainly deal with *topological changes* deriving from run-time failures and/or addition/removal of network resources, and do not explicitly take into account *varying traffic patterns*. On the other hand, most of the SI algorithms are explicitly designed to be adaptive to both topological and traffic variations.

Single-Path vs. Alternate- and Multi-Path. Single-path routing algorithms make use of a single-path at-a-time to forward traffic between two endpoints. The path is determined to be the best one available according to the considered performance metrics. Alternate path algorithms still make use of a single path but calculate and maintain also a backup path to be readily used in case of any problems or unavailability of the main reference path. Finally, multi-path algorithms discover, maintain, and use multiple paths to forward flows between the same source-destination pair. This allows to multiplex the traffic usually resulting in better failure resilience, utilization of network resources and higher throughput with respect to the other two mentioned strategies.

Flat vs. Hierarchical Organization. Flat routing protocols consider all nodes in the (sub)network as peers and maintain an entry in the routing table for each of them. This allows peers to discover best individual routes at the cost of transmitting a relatively large amount of control packets and maintaining large routing tables. Routing algorithms based on hierarchical organization, form logical groups of routers and organize them into areas, domains, and autonomous systems. This popular way of organizing the network, requires two types of routers, *interior routers*, which route traffic within a domain, and *exterior routers*, which route traffic between domains. Hierarchical organization requires significantly smaller routing tables with respect to a flat organization, requiring, in turn, smaller memory storage and less use of bandwidth to maintain routes.

Host vs. Router Intelligent. In host intelligent protocols a host determines the entire route to a destination and appends it to each packet header. This

way of proceeding is also known as *source routing*. The other routers in the system simply forward packets to the next hop specified in the header and in principle do not need to maintain up-to-date routing information for destinations not addressed by local sessions. On the other hand, in *next hop routing protocols* routing decisions are taken by the single routers ("router intelligent") that discover, maintain, and use paths on a per packet/flow basis.

Global vs. Local Representation. In routing protocols using a global representation, each node maintains a complete topological database of the network with the aim of constructing a network graph and apply (shortest) path finding algorithms on it. The popular class of link-state protocols exploits this strategy. On the other hand, protocols relying on local representations define the local routing policy on the sole basis of the use of local traffic and topology models. Distance-vector protocols make use of local representations. Link-state algorithms converge quicker, scale better, but require more CPU power and memory than distance vector algorithms. Therefore, they are more expensive to implement and support. SI protocols, which tend to simplicity, are usually based on local representations.

Deterministic vs. Probabilistic Decisions. Deterministic algorithms use a deterministic *selection rule* applied to the information contained in the routing table to decide next hops. Usually this results always in the greedy selection of the best routing alternative. Instead, probabilistic algorithms make use of a probabilistic selection rule. On the one hand this might result in locally sub-optimal choices; on the other hand, when multiple equivalent or comparable choices are available, the adoption of probabilistic routing selection will spread traffic across different concurrent paths implementing de facto a *multi-path* scheme and favoring *load balancing*. Clearly, a probabilistic scheme requires more computational and memory resources than a deterministic scheme to process each single packet and maintain all the necessary routing information [130]. A probabilistic decision scheme can be used also to forward control packets, not only data packets. In these cases, the probabilistic scheme can be exploited to provide a certain level of randomness in the way paths are discovered and set up. This is supposed to add robustness and flexibility to the routing system to better cope with the intrinsic network variability. As it will be shown later, probabilistic schemes for both data and control packets are widely adopted in SI algorithms.

Constructive vs. Destructive Routing Table Making. Constructive protocols start with an empty set of routes and incrementally add routes till the final routing tables are constructed. In contrast, destructive algorithms begin by assuming that all possible paths in the network are valid. That is, they assume that the network is a fully connected graph. Starting from this initial assumption, destructive algorithms incrementally gather information to cut paths that do not actually exist in the physical network [133]. Protocols based on strong exploratory/random strategies are

usually destructive, as it is the case of many SI protocols for wired networks. On the other hand, when the network topology is highly dynamic, for example, routes constantly appear and disappear as in the case of MANETs, the usual approach is the constructive one.

Proactive vs. Reactive Behavior. Reactive protocols gather routing information only in response to an event, usually one which triggers the need for new routes, such as the start of a data session toward a new destination or the failure of an existing route in use. In proactive protocols, routing information is constantly gathered, so that it is readily available when is needed. In the literature, proactive behavior is often associated to the fact that the protocol proactively defines and maintains routes toward all the possible destinations in the network. *Hybrid protocols* result from any combination of reactive and proactive behaviors. Usually all the protocols for wired networks offering best-effort services are proactive. QoS protocols are hybrid, with the reactive component addressing the QoS requests and the proactive component serving for both the QoS and the best-effort routes. Protocols for MANETs are rather uniformly distributed among the three different types of behaviors.

Proactive gathering of routing information can in principle permit to build sound statistical estimates of relevant aspects of the network dynamics that can be used in turn to *learn* and *adapt* with continuity the local routing policies. On the other hand, it is usually unfeasible to build sound statistical estimates when using a purely reactive strategy since there is not a continuity of information gathering. Clearly, an adaptive learning approach can result effective only if the network dynamics shows over time exploitable correlations at either the local or the global level, and does not hectically change with a high frequency.

Formal Guarantees vs. Emergent Behavior. Some algorithms come with formal guarantees concerning specific aspects of their behavior and performance. Properties that are particularly useful to be assessed regard: *failure resiliency*, establishment of *loop-less routes*, and *convergence* to an optimal route assignment. Fully deterministic algorithms designed according to top-down approaches have higher chances to enjoy verifiable properties than algorithms designed following a bottom-up approach and that make use of random components, which is often the case for SI algorithms. For this special class of algorithms, the resulting network behavior can be effectively categorized as "emergent", since its usually hard to provide a precise formal description of the expected network response and performance. On the other hand, also in the case of top-down design, the above mentioned properties can be usually asserted in special cases and only when steady stationary conditions are assumed, which is more the exception, rather than the rule, for network behavior.

4 From Insect Societies to Network Routing Protocols

Two specific classes of insect societies have inspired a relatively large volume of work in the specific domain of network routing: *ant* and *bee colonies*. More specifically, the ability of ant colonies to discover *shortest paths* between their nest and sources of food using a pheromone laying-following mechanism [60] has been reverse-engineered and put to work in the general optimization framework of the *Ant Colony Optimization (ACO)* metaheuristic [45, 44, 48]; see also Chapter 2 of this book. To date, ACO is a state-of-the-art metaheuristic for many problems in the domains of combinatorial optimization and network routing. More recently, the *communication* and *recruitment strategies* adopted for effective foraging within a beehive have inspired the development of some novel algorithms for routing problems.

In the following two subsections we discuss separately the general principles behind the ant- and bee-inspired approaches to network routing.

4.1 Shortest-path behavior in ant colonies and the Ant Colony Optimization metaheuristic

It has been observed that foraging ants in a colony can converge on moving over the shortest among different paths connecting their nest to a food source [60, 45]. The main catalyst of this colony-level *shortest path behavior* is the use of a *volatile* chemical substance called *pheromone*. While moving, ants lay pheromone on the ground and, at each step, they preferentially decide, with a random component, to locally move towards the adjacent areas marked by higher pheromone intensity. Shorter paths between the nest and the food source can be completed quicker and more frequently by the ants moving back and forth, and will therefore be marked with higher pheromone intensity. These paths will then attract over time more and more foraging ants, which will in turn increase the pheromone level of these paths, until there is convergence of the majority of the ants onto the shortest path(s). Fig. 1 illustrates this in a simple scenario with two possible paths of different length. At time $t = 0$ two ants leave the nest looking for food. No pheromone is present on the terrain. Each ant decides independently which way to go, and the random decision is biased by the amount of pheromone on each path. At $t = 1$ the ant following the shortest path has reached the food site. She decides to go back along the same path since it is marked with a higher amount of pheromone than the other, which has no pheromone yet. At $t = 2$ this ant is back to the nest, and a double amount of pheromone is now present on the shortest path. At $t = 3$ a new ant leaves the nest and selects the shortest path due to its higher concentration of pheromone, further reinforcing in this way its attractiveness compared to the longest path. In a short time, the great majority of the ants will convergence on moving on the shortest path.

The local intensity of the pheromone field encodes a spatially distributed *measure of goodness* locally associated to each moving decision. It is the result

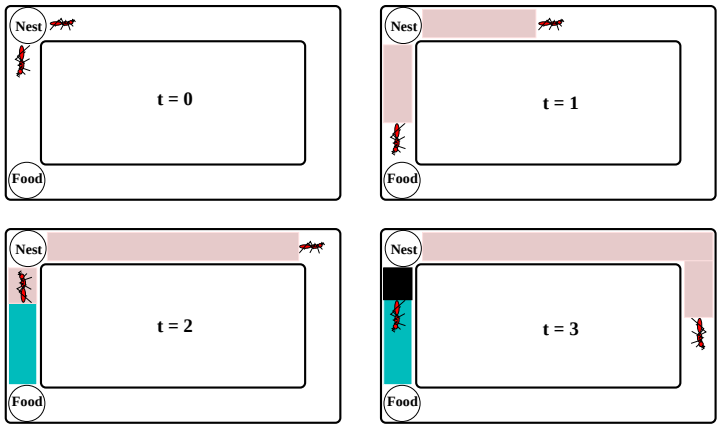

Fig. 1. Ant colony convergence onto the shortest path as a result of the pheromone laying-following behavior of individual ants (see the explanation in the text). A darker trail on a path indicates a higher amount of deposited pheromone.

of the repeated and concurrent *path sampling* experiences of the ants. In other words, it is the result of a *collective reinforcement learning process* [125, 24] happening at the colony level. This form of distributed learning and control based on indirect communication among *agents* (the ants) which locally modify the environment and react to these modifications leading to a phase of global coordination of the agent actions is called *stigmergy* [129]. In nature, ant colonies, as well as other social insects, make use of a variety of different pheromone signals for stigmergic communications. The different pheromones are secreted by different glands, and differ both in their chemical composition and their volatility. Recent studies have shown that this complex indirect signaling system based on *multiple pheromones* is efficiently exploited to react and coordinate in different ways to different stimuli in the environment [69]. For instance, the presence of a predator fuels the release of a danger-type of pheromone, while the discovery of a prey to be carried into the nest stimulates the generation of an intense but short-lived type of pheromone which is different from the long-lived pheromone laid for the exploitation of an abundant source of food. Pheromones can be not only *attractive*, as the ones described so far, but also *repulsive*. For instance, a branching leading to a bad route can be marked with repulsive pheromone to avoid its future selection.

Stigmergic coordination is one of the keys to obtain self-organized behaviors not only in ant colonies but more generally across social systems. When stigmergy is at work, system's *protocols* (interfaces) play a prominent role with respect to *modules* (agents) [22], which can be kept relatively *simple*. A good stigmergic model supplies robustness, scalability, evolvability, and allows to fully exploit the potentialities of the modules and of modularity. Stigmergic systems are paradigmatic examples of the swarm intelligence approach.

The ability of ant colonies to "solve" distributed shortest path problems using a number of minimalist agents and pheromone-mediate stigmergic communications has been exploited in the framework of the ACO metaheuristic, in which all the mechanisms at work in the ant colony shortest path behavior have been reverse-engineered to define a nature-inspired metaheuristic for the (distributed) solution of *generalized shortest path problems* in graph structures (notice that almost any network and combinatorial optimization problem can be formulated in terms of finding shortest paths in a graph [24]). The ACO metaheuristic features: *repeated path construction* by a distributed system of lightweight agents called *ants*, the use of a *stochastic decision policy* to incrementally construct each path by an ant that moves step-by-step from one node of the graph to an adjacent one, stigmergic communications among the ants through node-local stigmergic variables called *pheromone variables*, collective *stigmergic learning* of the pheromone variables, which represent the parameters of the decision policy, that is, which encode the expected quality of each decision about the next node to include into the path under construction.

The application of the ACO metaheuristic to network routing is quite straightforward. This results both from the intrinsic distributed architecture of the metaheuristic and from the fact that the problem of defining optimized routing paths in a network environment can be configured as a particular instance of a shortest path problem, with the weights of the edges being dynamic values depending on bandwidth, propagation delay, and input traffic (whose characteristics are usually unknown with precision in advance).

4.2 Useful Ideas From Honey Bee Colonies

More recently than ant colonies, also honey bee colonies have attracted a strong interest as a potential source of inspiration for the design of optimization strategies for dynamic, time-varying, and multi-objective problems. Bee colonies show structural characteristics similar to those of ant colonies, such as the presence of a population of minimalist social individuals, and must face analogous problems such as distributed foraging, nest building and maintenance, etc. Bees utilize a sophisticated communication protocol that enables them to communicate directly through bee-to-bee signals and when required, similar to ants, use stigmergic feedback cues for bee-to-group or group-to-bee communication. In these two classes of insects, communication and cooperation is realized according to radically different modalities due to the different nature of these insects (ants mainly walk, while bees mainly fly). In particular, while in the case of ants communication is achieved via a pheromone trail that is laid on the ground while walking, in the case of bees it is a form of visual communication that plays an equivalent role. In the following we briefly point out and discuss the main mechanisms at work in a bee colony which have found their application in the design of routing algorithms.

Adaptive and age-related division of labor

A honey bee colony consists of morphologically uniform individuals with different temporary specializations [114]. The benefit of the organization is an increased flexibility to adapt to the changing environments. For instance, a nectar forager can become a water forager if the colony is running out on its water supplies. More specifically, in honey bees *division of labor* is mainly related to *age*: workers of different ages specialize in different tasks (this phenomenon is called *age polyethism* or behavioral development). Workers typically perform brood rearing for the first week, engage in other hive maintenance duties (wax secretion, guarding, undertaking, nectar processing) when they are "middle-aged" (2-3 weeks old), and switch to foraging and colony defense when they are about three weeks old. These phases can be adaptively modified in response to the alteration of colony conditions.

Communication inside the colony and worker recruitment

As in the ant case, also in a bee colony *foraging* is critical aspect for the survival of the colony and is executed in a *fully distributed* and *competing* way. Foraging bees constantly leave the hive searching for new sources of nutrient, bring the nutrient back to the hive, and try to recruit other bees to exploit the food site found by competing with each other during the recruitment process. Foragers announce a food source of interest to their fellow foragers by doing a dance on the dance floor inside the hive [136, 137]. This dance is termed *waggle dance*. It is a particular figure-eight dance that encodes the direction of the food source in the angle from the sun, and the distance in the duration of each waggle-run [114]. If the distance is very short the waggle dance resembles a *round dance*. Foragers respond to the waggle dance with a strong preference for choosing nearer food sites over distant ones in order to increase the net energetic efficiency of the colony. The waggle dance is a direct form of *agent-to-agents communication*.

Nectar foragers, upon return to the hive, sometimes also perform across the hive a quite strange dance termed *tremble dance*. The tremble dance means that the forager has found a rich food source but upon return to the hive, after a certain threshold time, she could not find a food-storer bee to give her nectar. This suggests that the message of the tremble dance is to stimulate the bees inside the hive to increase and/or to switch to nectar processing activities, and to inhibit the outside foragers from recruiting additional bees. Basically the tremble dance is intended to activate behaviors that keep a colony's nectar processing rate matched with its nectar intake rate.

Stochastic selection of food sites

The unemployed foragers refrain from extensively surveying the dance floor to identify the best food site. On the contrary, they observe maximally two or three dances on the dance floor and then decide to follow the indications of one of them according to a *stochastic rule*. As a result, a colony *distributes*

its foraging force on multiple food sites such that when one rich food site has been almost fully exploited the colony is already exploiting other sites [114]. In this way an effective *balancing* between exploitation and exploration is automatically obtained. Sumpter [124] has developed a formal agent-based model using process algebra for the foraging behavior of honey bee colonies which provides some useful insights about the colony-level strategy for the distribution of the exploitation activities.

5 Routing Protocols Based on Ant Colony Optimization

5.1 General Structure and Properties of ACO Routing Protocols

The main characteristic of an ACO routing algorithm [24, 38] consists in the continual acquisition of routing information through *path sampling* and *discovery* using small control packets, the ants. The aim is to adaptively learn statistical estimates of the quality (e.g., expected end-to-end delay) of each local routing choice. The ants are generated concurrently and independently by the nodes, with the task to try out a path to an assigned destination. An ant going from source s to destination d collects information about the quality of the path, and, either on the way to d or while retracing its way back from d to s, it uses this information to update the routing tables at intermediate nodes, reinforcing the good paths. In other words, the repeated path sampling and consequent reinforcement of good routing decisions, is a form of *distributed reinforcement learning based on stigmergy* (e.g., see [100, 24]). The routing table at node i is derived from the so-called *pheromone table* $\mathcal{T}^i$, which contains for each destination d of interest a vector $\bar{\tau}_d$ of real-valued entries τ_{nd}, one for each node n in the reachable neighbor of i, indicated hereafter with $\mathcal{N}^i$. These entries, which are the *pheromone variables*, are a *local* measure of the goodness of going over the neighbor n on the way to d. They are continually updated according to the quality of the paths *sampled* by the ants. The repeated and concurrent generation of ants results in the availability, at each node, of a *bundle of paths*, each with an estimated measure of quality based on pheromone. The information from the pheromone tables is usually combined with additional *heuristic information* η not depending or derived from ant sampling activities, to obtain the *selection probabilities* p which are used by the ants to find their way to the assigned destination d: at each node i they stochastically choose a next hop $n \in \mathcal{N}^i$ giving higher preference to those next hops which are associated with higher p_{nd} values, which are calculated as some function f of both pheromone and heuristic values, $p_{nd} = f(\tau_{nd}, \eta_{nd})$. The heuristic values have the same structure as the pheromone ones and associate to each pair (next hop, destination) a heuristic measure of goodness. For instance, the number of packets waiting on the queue for link $i \rightarrow n$ can be used as a local measure of the goodness of using that link. However, not all the implementations make use of a heuristic correction to the pheromone values to derive the selection probability values.

In the case of connectionless networks, packets are usually routed more or less in the same way as the ants: packets are routed *stochastically*, choosing with a higher probability those links associated with higher pheromone/ant-routing values. This way data for a same destination are adaptively spread over *multiple paths* (but with a preference for the best paths), resulting in *load balancing*. In the case of connection-oriented networks, spreading can be done at the level of virtual or physical circuits. For both data packets and circuits, mechanisms are usually adopted to avoid low quality paths, while ants are more explorative, so that also less good paths are occasionally sampled and maintained as *backup paths* for failures or sudden congestion. In this way *path exploration* is kept separate from the use of paths by data. If enough ants are sent to the different destinations, nodes can keep up-to-date information about the best paths, and automatically adapt their data load spreading.

Referring to the classification features of Sect. 3, ACO implementations for routing usually show the following characteristics: (i) they are all adaptive, with a special focus on traffic patterns, (ii) they usually provide and use multiple paths, (iii) they are mostly based on a flat organization, (iv) router-intelligent schemes are the most adopted ones, (v) global representations are barely used since the approach in a sense emphasizes simplicity and locality, (vi) probabilistic exploratory decisions are an integral part of all the implementations, (vii) they adopt either a constructive or a destructive approach depending on the network type, (viii) the majority of the implementations follow either a proactive or a hybrid scheme, and make use of some form of incremental learning to continuously adapt over time the routing tables to network changes, (ix) usually these algorithms come with no or little formal guarantees apart from some guarantees of probabilistic convergence to the optimal policy under stationarity, and the probabilistic guarantee that a packet following a loop will be routed out of the loop in a short time.

The first ACO routing algorithms were developed at the beginning of the second half of the 90's and were designed for wired networks: *AntNet* [29] for connectionless IP data networks and *ABC* [113] for circuit-switched telephone networks. A number of other ACO implementations for different routing problems have been developed since then. The majority of these subsequent implementations have based their design on the general features and architecture of either AntNet or ABC. Therefore, in the following we give a special attention to these two algorithms that can be considered as the main reference templates for ACO routing implementations and can help to understand the common architecture and characteristics of most of the other implementations.

In [24, 38, 37] Di Caro et al., starting from the observation of the existence of a core set of features common to most of the ACO-derived algorithms for routing, defined the *Ant Colony Routing* (ACR) framework. ACR includes basic ACO concepts but at the same time extends them with notions from the domains of reinforcement learning [125], multi-agent systems, and autonomic networking [73], and specializes them for the specific class of network routing problems. The ACR framework is intended to provide the basic guidelines

for the design of novel adaptive protocols for routing in modern dynamic networks. Because of lack of space, we are not going to discuss here the ACR framework. However, it is worth to point out that ACR explicits the two main mechanisms for *monitoring* and *learning* which are at the core of most of the routing algorithms derived from ACO: node-local monitoring of traffic dynamics for *inductive learning* of congestion and routing information, and non-local *sampling/probing* of full paths by using ant agents that implements a combination of *active learning* and *Monte Carlo learning* [125] strategies. The use of these techniques is in some sense not new to the field of networking. The use of inductive learning traces back to the work on learning automata of Narendra et al. [90, 92], while active probing has been widely used to estimate characteristics of network paths (e.g., [68]). However, the way their are combined, implemented, and used in ACO-routing, and, more generally, in ACR, is innovative and highly effective.

The interested reader can find additional definitions, discussions, and analysis concerning the application of ACO to routing in [24, 38, 121, 29].

5.2 AntNet: The Main Reference Algorithm for Connectionless Networks

AntNet (1997) [29, 24, 30, 28, 25] was proposed by Di Caro and Dorigo for dynamic best-effort routing in wired IP networks such as the Internet. The algorithm is explicitly designed to provide *traffic-adaptive routing*. Topological changes are not explicitly considered, such that route breaks due to link failures are only dealt with implicitly by reacting to the increase of the number of data packets waiting in the queue of the broken link. A flat network organization with router-intelligent hosts is assumed. Informally, the behavior of AntNet can be summarized as follows.

At the beginning of the operations routing tables are initialized with uniform equal values for all the neighbors, basically adopting a destructive approach. They are then adapted over time as a result of the ant-based activities. At regular fixed intervals and concurrently with data traffic, ant agents are *proactively* and independently launched from each network node s towards destination nodes d which are chosen following a *random proportional* selection that favors the locally most requested destinations, or implementing with a very small probability a *random uniform* selection. These ants are called *forward ants*. A forward ant is a sort of random experiment aimed at exploring the network searching for a *minimum delay path* connecting ant's source and destination nodes, and gathering at the nodes information about the end-to-end delay for the followed path. Ants, once generated, act as *autonomous agents*. They communicate in an indirect, stigmergic way, through the information they locally read from and write to the nodes in three data structures: the *pheromone table* $\mathcal{T}$, the *parametric statistical model* $\mathcal{M}$, and the *data routing table* $\mathcal{R}$, that together define the routing information database locally available to issue routing decisions (see also Fig. 2).

The pheromone table is a stochastic matrix which is used by the ants as a routing table. Each pheromone estimate $\tau_{nd} \in \mathcal{T}^i,\ n \in \mathcal{N}^i$, is the result of the continual path sampling and learning activities of the ants, and is related to the inverse of the estimate of the expected minimum time to reach d. τ's values for the same destination d are normalized to one ($\sum_{n \in \mathcal{N}^i} \tau_{nd} = 1$). This allows to treat the pheromone values as probabilities and better evaluate the relative goodness of each neighbor. $\mathcal{M}^i$ is a parametric statistical model for the traffic and delay situation on the paths to reach the different destinations. $\mathcal{M}^i$ is a vector of N triples (μ_d, σ_d^2, W_d), with N being the number of destinations. μ_d is the sample exponential mean of the ants' traveling time to reach d, σ_d^2 is its variance, and W_d is the best end-to-end time observed during the last window of w ant samples. Finally, the data routing table $\mathcal{R}^i$ is the stochastic matrix used for routing data packets. It is derived from the pheromone table by an exponentiation and renormalization process that assigns to the best routes much higher selection probabilities than in the case of the pheromone table. This is because the ants are supposed to explore, while the data packets are supposed to exploit at best the paths found by the ants.

Forward ants *simulate data packets*. They move hop-by-hop towards their destination making use of the same priority queues used by data packets, experiencing in this way the same delays. During its journey to d, a forward ant stores in its memory the traveling time $t_{i \to j}$ between each hop $i \to j$ and the identifiers of the visited nodes along the followed path $\mathcal{P}_{s \to d}$. At each intermediate node i, a *stochastic decision policy* $\pi_\epsilon(\mathcal{T}^i, \mathcal{L}^i, \mathcal{P})$ is applied to select the next node $n \in \mathcal{N}^i$ to move to, where $\mathcal{N}^i$ is the set of neighbors of i.

The *selection probability* p_{nd} assigned to each neighbor $n \in \mathcal{N}^i$ is a measure of the goodness, relative to all the other $j \in \mathcal{N}^i, j \neq i$, of using the neighbor as next hop for d as final destination. p values are calculated considering a combination of: (i) the pheromone value τ_{nd} which is the result of the continual, long-term path sampling and learning activities of the ant agents, (ii) the length in bytes to be sent of the link queue $l_n \in \mathcal{L}^i$ associated to n, which is a *heuristic* instantaneous measure of congestion of the path going through n, and (iii) the list of the visited nodes stored in the ants' memory, which is used to avoid loops. More precisely, each p_{nd} is defined as:

$$p_{nd} = \frac{\tau_{nd} + \alpha l_n}{1 + \alpha(|\mathcal{N}^i| - 1)} \tag{1}$$

if $n \notin \mathcal{P}_{s \to i}$, zero otherwise. In practice, with this formula, the selection probability of a next hop is calculated as the weighted sum of the estimate τ, which is the result of a continual process of incremental learning, and the instantaneous quality estimate l. Both τ and and l values are scaled between 0 and 1, in order to be summed consistently. $\alpha \in [0, 1]$ determines the relative importance of the long-term versus the instantaneous view of the goodness of each next hop decision. The denominator is just a normalization factor.

Once arrived at destination, the forward ant becomes a *backward ant*, which is *source-routed* to s: it goes back to its source node by moving along

the same path $\mathcal{P}_{s\to d} = [s, v_1, v_2, \ldots, d]$ as before but in the opposite direction. For its return trip the ant makes use of high priority queues to quickly retrace the path and update the routing information.

Arriving from neighbor j, at each visited node $i \in \mathcal{P}_{s\to d}$ the backward ant updates, for the choice of j as next hop, the routing information related to each node $\delta \in \mathcal{P}_{i\to d}$ visited by the forward ant when traveling from i to d. Basically, each node δ is considered as an intermediate destination. The backward ant first *evaluates* the goodness of the followed path and of its sub-paths, and then uses this evaluation to update the local routing information. Path evaluation is done by comparing the traveling times experienced along the path with the expected traveling times maintained in the statistical model $\mathcal{M}^i$. From the evaluation process, a path *reinforcement* value $r \in [0, 1]$ is defined as:

$$r = c_1 \frac{W_\delta}{T_{i\delta}} + c_2 F(T_{i\delta}, \mu_\delta, \sigma_\delta^2, W_\delta), \tag{2}$$

where c_1 and c_2 are weighting factors, $c_1 + c_2 = 1$, and F is a real function accounting for the statistical dispersion of the sampled values. In practice, the sampled path (and sub-paths) gets a reinforcement proportional to how good is the traveling time $T_{i\delta}$ just experienced by this ant compared to what has been observed in the recent past. At the visited nodes i, r is used to update the pheromone entries as follows. The path to each "destination" δ going through the used neighbor j is reinforced, while, by normalization, the goodness of all the other alternatives is proportionally decreased:

$$\begin{aligned} &\tau_{j\delta} \leftarrow \tau_{j\delta} + r(1 - \tau_{j\delta}), \\ &\tau_{k\delta} \leftarrow \tau_{k\delta} - r\tau_{k\delta}, \quad \forall k \in \mathcal{N}^i, k \neq j. \end{aligned} \tag{3}$$

Fig. 2 shows the data structures used by the ants at the nodes, and illustrates the two core phases of in the AntNet operations: the decision step of the forward ant and the update process executed by the backward ant.

Once the ant has returned to its source node, it is removed from the network. *Data packets* are routed according to a stochastic decision policy similar to that of the ants but based on the information contained in the local data routing table $\mathcal{R}$, which is derived from the pheromone table used to route the ants preferring the best paths. In this way, data traffic is concurrently spread over the best available *multiple paths*, resulting in an optimized utilization of network resources and in automatic *load balancing*.

AntNet-FA [32, 24] (also known as *AntNet-CO*) is a minor but quite effective improvement of AntNet: also forward ants make use of high priority queues. In this way, forward ants quickly get to the destination, and do not need to carry traveling times, it is the backward ant that calculates incrementally the trip times while traveling backward. Coming from neighbor n, at node i the backward ant estimates the time it would be necessary to cross the link $i \to n$ by looking at the number of bytes waiting in the l_{in} queue. The link crossing time T_{in} is obtained on the basis of a queue depletion model:

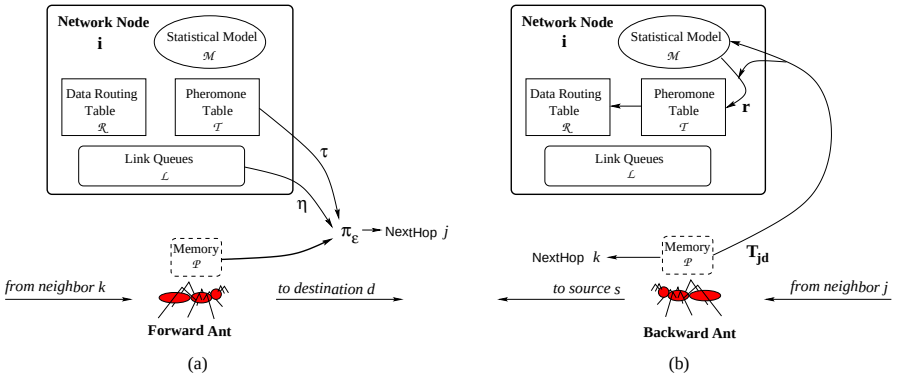

Fig. 2. The two core phases of AntNet shown at a node $i \in \mathcal{P}$ for an ant generated in s and targeted to d: (a) the decision step of the forward ant, and (b) the update and move step of the backward ant. The arrows serve to visualize from which data structures the ant gets the information to decide the next step during the two phases, and the logical sequence of updating steps happening during the backward phase.

$$T_{in} = \frac{l_{in}}{b_{in}} + d_{in}, \tag{4}$$

where b is the link bandwidth and d is its propagation delay. The adopted model is simple but also quite reliable. AntNet-FA's strategy on one side permits to calculate source-destination trip times which are more up-to-date than those used by AntNet's backward ants, and on the other side it allows a quicker gathering and spreading of routing information. This is a clear advantage in the case of large topologies and quickly changing input traffic.

AntNet's authors have evaluated their algorithm on the basis of a relatively large number of *simulation experiments* using a custom network simulator. The algorithm has been tested on a variety of different scenarios based on different topologies with number of nodes ranging from few units to 150, and considering UDP traffic patterns with different geographical and generation characteristics. *Throughput*, *90th percentile of packet delays*, and *routing overhead* have been chosen as performance indices. The reported experiments show that AntNet robustly outperforms in terms of throughput and delay several different dynamic state-of-the-art algorithms: *Q-routing*[9], *PQ-routing* [19], *Shortest Path First (SPF)* [74], *Dynamic Bellman-Ford* [115], and *OSPF*. The improvement in performance is achieved without incurring in larger routing overhead. Moreover, AntNet-FA outperforms AntNet, with the difference becoming larger with the increasing of the network size.

5.3 ABC: The Main Reference Algorithm for Connection-Oriented Networks

Schoonderwoerd et al. (1996) [113, 112] were the first to apply the ACO ideas to routing and load-balancing problems in networks. More precisely,

they considered a telephone network in which the connection between sender and receiver is explicitly established by reserving a virtual circuit. In their network model, each node is a crossbar switch and can handle only a limited number of simultaneous call. Connection links are seen as full-duplex channels with infinite capacity. Therefore, network bottlenecks are nodes' capacities. This means that the network is *cost-symmetric*: the congestion status over an end-to-end path is the same in both directions since it only depends on the spare connection capacity at the nodes (e.g., see [57]). The proposed routing algorithm, named `Ant-based control (ABC)`, aims at distributing the calls over multiple switches (i.e., *load balancing*) to minimize the number of calls that cannot be routed because of congestion.

ABC and AntNet share the same general organization and principles. The main differences between the two algorithms are for aspects deriving from the differences existing between the two different network scenarios that have been addressed. In ABC ants move over a control network isomorphic to the one where the calls are established. In the adopted model the system evolves synchronously in discrete steps. Next hops are selected according to a random proportional or random uniform rule, as in AntNet, but taking into account only pheromone values, no heuristic correction is used. Arrived at a node, an ant waits ΔT steps defined as a function of the spare node capacity ΔC,

$$\Delta T = Ke^{-a\Delta C}, \tag{5}$$

with K and a real constants, $K \gg a$, and increases in this way its *age*. This is equivalent to what happens in AntNet, where forward ants wait in the local data queues, with a consequent increase in their traveling time. Equivalently, the age is used in ABC to asses the quality of the ant path: an old ant is associated to a congested path. Pheromone entries are updated using the ant age T as follows. If s is the source and d the destination node of a traveling ant, after crossing the control link $i \rightarrow j$ the probabilistic pheromone table $\mathcal{T}^j$ at node j is immediately updated using the total ant age T. A reinforcement r inversely proportional to T is assigned to the normalized entry τ_{is} in $\mathcal{T}^j$:

$$r = a/T + b, \tag{6}$$

where a and b are small constants dependent on network characteristics. The updating formula for the τ values is the same as in AntNet (Eq. 3). The main difference with AntNet in this respect consists in the fact that the pheromone table is updated during the forward journey in the backward direction of the source node s. This way of proceeding is justified by the fact that the network is cost-symmetric, such that the cost (level of congestion) of a path is the same in both directions. Therefore, at node j the ant age is a sound measure of the quality of the reverse ant path $j \rightarrow s$. In ABC ants do not need to retrace the path backward. Calls are routed according to a deterministic greedy policy that always selects the best next hop. If the destination can be reached, a circuit is established and the call can happen.

ABC's performance has been tested in simulation considering the real topology of the backbone of the British Telecom (BT) telephone network and a number of different call patterns. Reported results show that ABC outperforms an agent-based algorithm developed for BT by Appleby and Steward [1] and reacts better to changes in traffic.

5.4 Algorithms for Wired Connectionless Networks

In this section we review the main work concerning the application of AntNet, ABC, and, more in general, ant colony ideas, to wired best-effort routing in connectionless networks such as the Internet.

Subramanian, Druschel, and Chen (1997) [123]: `Uniform Ant Algorithm`

The authors consider generic cost-aysmmetric networks and provide an analysis of two algorithms, one is based on ABC, and the other is a very simple one that makes use of on so-called *uniform ants*. In both algorithms, ants make routing table updates in the *reverse direction* of their motion: arriving at node j from node i, an ant originally launched from s updates the $\tau_i s$ entry of j's routing table using some measure c_{ji} of link cost calculated in j. The difference between the two algorithms consists in the fact that uniform ants wonder in the network with no specific destination and make next hop selections blindly, without relying on pheromone. The core idea behind uniform ants is that *simple unbiased exploration* is a mean to adapt to any change in the network, especially failures. Since they sample all the paths with equal probability, this results in setting up a fully *multi-path* system. Moreover, the fact that they have no destination, make them potentially useful also in ad hoc networks in which node identifiers are not globally known in advance. The authors provide some theoretical proofs of asymptotic convergece of the two algorithms under stationary link costs. Simple simulation experiments considering small topologies show that the two approaches are more or less equivalent and comparable to simple link-state and distance-vector algorithms. The downside of the simple and general mechanism of uniform ants consists in the fact that its efficacy and efficiency is expected to dramatically decreases with the increase of network size. In some sense, the core idea behind ACO is precisely to find optimized ways to implement biased exploration and/or deal with failures, rather than relying on blind mechanisms.

Heusse et al. (1998) [66, 67]: `Cooperative Asymmetric Forward` *(CAF)*

CAF extends ABC's strategy for step-by-step updating in cost-asymmetric networks. In CAF, when a data packet arrives at node i, the arrival time t_i is written in the packet. After arriving at j from i at time t_j, the total time elapsed to go from i to j, $t_{ij} = t_j - t_i$, is written in j . An ant hopping from j to i reads the t_{ij} information in j and moves it to i, where it is used to update the local estimate for the time to travel from i to j. Since the ant is

doing this for all the nodes along its path, the estimate of the traveling cost from i to all the nodes the ant has visited so far can also be updated and used to update step-by-step the pheromone tables in the direction opposite to the ant motion, as in ABC. Clearly, if an ant arrives some time after the data packet, the information carried back by the ant might be out-of-date. The authors tested CAF under some static and dynamic conditions, using the average number of packets waiting in the queues and the average packet delay as performance measures. In [66] they compared CAF to an algorithm very similar to an earlier version of AntNet [26] and to Q-routing. Results were encouraging and under all the test situations CAF outperformed its competitors. In [67] the effectiveness of the approach for load balancing was successfully compared to more classical approaches.

Van der Put and Rothkrantz (1998) [132, 131]: `ABC-backward`

ABC-backward is designed as a combination of the basic ABC structure and formulas with the forward-backward updating strategy of AntNet. The authors have experimentally verified that ABC-backward has a better performance than ABC on both cost symmetric and cost asymmetric networks.

Oida and Kataoka (1999) [94]: `DCY-AntNet, NFB-Ants`

The authors improved an earlier version of AntNet [26] in which the heuristic term based on the instantaneous status of the data link queues was not included into the selection formula (Eq. 1). Without this dependency on the status of the queues, AntNet will suffer from what is termed *stagnation* in the ACO jargon: once the pheromone value τ_{nd} of any next hop link of a neighbor reaches 1 the routing tables get "locked". In ACO algorithms for combinatorial problems this problem is bypassed by applying at each time step t a sort of *pheromone evaporation* to all pheromone entries: $\tau_{nd}(t+1) = \rho\tau_{nd}(t),\ \rho \in [0,1]$. The use of an evaporation mechanism allows to keep good levels of exploration at any time. The authors of [94] modified pheromone table updating rules to avoid the locking behavior. Their algorithms, `DCY-AntNet` and `NFB-Ants`, upon comparison with the considered earlier version of AntNet performed much better under challenging situations.

Doi and Yamamura (2000) [40, 41]: `BNetL`

These authors also proposed a few additional heuristics to avoid the same locking problem addressed by Oida and Kataoka, but this time considering *AntNet-FA*, which is actually lock-free. Consistently, their algorithm showed a performance equivalent to that of AntNet-FA.

Baran and Sosa (2000) [3]: Improving AntNet-FA

These authors have introduced several modifications to AntNet-FA: (i) instead of starting from a uniform pheromone distribution among all the available next hops for all destinations, for the destinations coinciding with the actual

neighbors, pheromone is explicitly initialized to give a much higher selection probability to the shortest, one-hop, route; (ii) assuming the existence of a mechanism that can locally detect and notify a link failure, the pheromone values for the next hop associated to the currently unavailable link are explicitly set to zero, this makes the algorithm explicitly *failure-resilient*; (iii) so-called *uniform ants* adopting a uniform random decision policy like in [123], are introduced to avoid the stagnation effect (however, as mentioned above, AntNet-FA does suffer from this, and therefore the introduced mechanism just helps to increase *exploration*); (iv) for the purpose of better exploiting the best paths, regular ants implement *greedy deterministic decisions* instead of random proportional ones, on the other hand, this reduces exploration (counterbalancing the effect of using uniform ants), and raise the probability that ants and data packets get trapped in long-lasting loops; (v) in order to limit routing overhead, the number of ants concurrently active in the network has been arbitrarily limited to four times the number of the links, unfortunately this can also impair the responsiveness of the algorithm and it is not precisely controllable in a distributed way.

Fenet and Hassas (2000) [55, 56]: Load balancing system

This work aimed at developing a novel multi-agent system for *multiple-criteria load-balancing* on a network of processors. The proposed system, which consists of both static and mobile agents, shows general characteristics similar to those of the previously mentioned ACR framework.

Michalareas and Sacks (2001) [86]: Deterministic simplified AntNet

In this work the authors have replaced the stochastic decision policy of AntNet with a *deterministic greedy policy* and did not use the heuristic based on queue lengths. This deterministic version of AntNet has been compared in simulation to OSPF on small tree, ring, and star topologies, and by considering FTP traffic using TCP Tahoe. According to the reported results, under stationary traffic conditions both the algorithms show equivalent performance.

Kassabalidis et al. (2002) [72]: `Adaptive-SDR`

This algorithm is derived from AntNet but makes use of a *hierarchical organization* by structuring the network into clusters using a centralized *K-means* algorithm. Once the partition process is completed, the algorithm maintains inter-clustering and intra-clustering routing tables at each node. *Multiple colonies* of ants are used to discover and maintain these different routing tables. In this manner the number of ants which need to be generated is significantly reduced because a node only maintains routes to the nodes inside the cluster and not to all the nodes in the network. The authors have compared Adaptive-SDR with a custom, non-standard, implementation of AntNet in which data are routed using a deterministic greedy policy, and with OSPF and RIP. Reported simulation results show that Adaptive-SDR

achieves the best results regarding throughput and average delay. The experiments were conducted on 16 and 48 nodes network topologies using the *NS-2* simulator [93]. The same authors provided in [71] a brief overview of swarm intelligence for routing, basically presenting ACO approaches.

Yang et al. (2002) [155]: AntNet on a real network

Differently from all previous works which were based on simulation, these authors implemented and studied AntNet on a *real network*, a 5-nodes LAN of Windows-based machines using the TCP/IP protocol. To shorten implementation time, the algorithm was actually implemented at the *application layer*, and not at the network layer. The authors made a study of the relative merits of different ways to define the reinforcement parameter r (see Eq. 2), which is central to the stable operation of the algorithm. They observed that the case of *constant reinforcements* leads to slow but dependable performance, whereas *adaptive reinforcements* might bring better performance but appear to be sensitive to the window length w used for statistics.

Doi and Yamamura (2004) [42]: Loop-free AntNet

This work addresses two important aspects that have been neglected in most of the other mentioned works: (i) the fact that the Internet has a hierarchical structure and shows power-law properties regarding its topology, and (ii) a routing algorithm should provide some guarantees in terms of being loop-free. The authors proposed a loop-free variant of AntNet-FA in which forward ants explicitly avoid to consider for next hop selection all the nodes previously visited. Both the original AntNet-FA and the loop-free variant have been tested on a set of hierarchical, scale-free, Internet-like topologies, and found that the topological characteristics have a significant impact on the relative performance of the two algorithms.

Lang, Zincir-Heywood, and Heywood (2006) [78, 77]: AntNet vs. Distributed Genetic Algorithms

The authors have benchmarked AntNet and their `GA-agent` (2002) [79], based on a *distributed genetic algorithm* architecture, against several dynamic scenarios considering the 56-node topology of a former backbone of the NTT Japanese company. AntNet was found to be able to deliver the best routing performance providing that complete and up-to-date global information on the number and identifiers of the reachable network nodes is given in input to the algorithm. On the other hand, the GA-agent algorithm, which does not require the a priori global knowledge, is shown to provide a performance which is intermediate between that of AntNet with and without global information.

Castrate et al. (2006) [135]: AntNet on a real network

These authors have implemented AntNet on a *physical network* of 5 routers and 2 hosts. The authors ran extensive tests to tune AntNet's parameters

and extend and modify the basic algorithm to make it working properly in a physical network. AntNet's performance has been compared to OSPF for throughput and failure adaptivity. In terms of throughput, AntNet largely outperformed OSPF in all the tested situations. On the other hand, since AntNet has not a built-in mechanism to deal explicitly with topological failures, it recovers to failures slower than OSPF. The authors added a simple mechanism to overcome this problem, and were able to obtain significantly better performance than OSPF also with respect to topological failures.

Dhillon and Van Mieghem (2007) [23]: AntNet performance analysis

This work aimed at getting a deeper understanding of the properties of AntNet. The authors have made a performance analysis of AntNet comparing it with a centralized Dijkstra's shortest path algorithm. The reported simulations show that the performance of AntNet is in general comparable to Dijkstra's algorithm. However, under varying traffic loads AntNet adapts better to the changing traffic and outperforms shortest path routing.

Gadomska and Pacut (2007) [58]: AntNet with TCP and UDP

It is well known that the TCP, the Internet transport protocol, can show performance degradation in case of the arrival of out-of-order packets. This might happen because of packet losses, or when an adaptive multipath routing algorithm is used at the network layer, or when the network is undergoing repeated topological modifications. In this work, the authors have studied the effect on performance of using an adaptive multipath routing algorithm like AntNet at the network layer, together with either UDP or TCP at the transport layer, while the majority of the previously mentioned works are all based on the use of UDP. The authors have run a number of simulation experiments using different realistic network topologies, input traffic, and TCP implementations. Reported results show that while TCP sets higher demands than UDP on the adaptation processes, it is still possible to improve network performance with the use of an adaptive algorithm at the routing layer. In some cases the use of TCP can even improve adaptation time.

5.5 Algorithms for Wired Connection-Oriented Networks

In this section we review the main work concerning the application of ACO ideas to wired connection-oriented networks such as telephone networks and IP networks using virtual circuits (but not explicitly providing QoS).

Di Caro and Dorigo (1998) [31]: `AntNet-FairShare (AntNet-FS)`

Starting from their AntNet-FA, the authors have derived a novel model for *fair-share* routing and flow control in *virtual circuit* networks. In their model, for each flow a virtual circuit is allocated and bandwidth is reserved. However, the allocated bandwidth is not that requested by the session, it is the maximum bandwidth that can be provided at the moment the session is active and

on the basis of a fair-share distribution of the bandwidth among the users. In AntNet-FS, on-demand mechanisms for session setup are added to the usual proactive ant generation. On the arrival of a new traffic session, a *forward setup ant* is *reactively* generated to find and reserve one or more paths for the session. During its journey toward the destination, it behaves like an AntNet-FA's forward ant, except for the fact that, if multiple equally good alternatives exist at a node, the ant is replicated and sent over all the equally good next hops. Moreover, the ant read from the nodes the value of their residual available bandwidth. The first setup ant arriving at the destination goes back and allocates a virtual circuit with a reserved bandwidth that equals the minimum, bottleneck, bandwidth which is available along the path, and that does not exceed the bandwidth needed by the session. Further setup ants arriving at destination are allowed to go back and add virtual circuits only if their trip time is comparable to that of the first ant and their path is sufficiently disjoint from those of the circuits allocated so far. Each session is forced to limit its data generation to not exceed its reserved bandwidth. On subsequent session arrival/departure, bandwidth allocation is dynamically recalculated and the sessions are notified in order to adjust their data rates.

White, Pagurek, and Oppacher (1998) [153, 152, 154]: ACO, pheromone evaporation, and genetic algorithms

These authors described several models and implementations for routing and path finding based on ACO [153, 152] or, more generally, on swarm intelligence [154]. The systems they proposed have an architecture which is very similar to the one of AntNet-FS [31] (described in Sect. 5.6) but make use of pheromone updating formulas which are adapted from *Ant System* [47], one of the earlier ACO implementations for the traveling salesman problem. In particular, they imported from Ant System the notion of *pheromone evaporation* (see also Sect. 5.4) to sustain path exploration. The authors considered static and dynamic scenarios, as well as centralized and distributed ones. They conducted experiments on small topologies, and results show that the proposed algorithms are able to compute shortest paths in the considered situations. In [152] they used a *genetic algorithm* to dynamically adapt the parameters weighting the relative importance of pheromone and heuristic correction at routing decision time. The use of the genetic algorithm in their `ASGA` routing algorithm resulted in improvement of the performance.

Bonabeau et al. (1998) [8]: ABC and dynamic programming

This work extended ABC with *smart ants* derived from dynamic programming: an ant launched from s, at node i updates the pheromone values for all nodes visited during its trip, rather than just for the source node, as in ABC. That is, all the sub-paths of the $\mathcal{P}_{i \to s}$ path are updated. This is the same strategy adopted in AntNet and in many other algorithms. Compared to ABC ants, smart ants have a more complex behavior but on the other hand, a better performance is achieved with less agents.

Sandalidis, Mavromoustakis, and Stavroulakis (2001) [111, 110]: Improving ABC with anti-pheromone

In their first work [111], these authors have studied the behavior of ABC on a few different network topologies and have confirmed the earlier results published by the authors of ABC. More recently, in [110] the same authors further improved the original ABC: if the age of an ant arrived at node i is greater than the maximum age calculated so far at i, then the pheromone entry related with the ant path is *decreased* instead of being increased. This is a form of so-called *anti-pheromone* similar to the repulsive pheromone used by ants in nature to block unfavorable paths (see 4.1): in the presence of an experimental evidence that a sampled route is not good compared to other available routes, its probability of being selected is explicitly decreased. In the large majority of the other ACO implementations, after being sampled, the selection probability of a route is always increased. The performance of the algorithm has been compared to that of ABC for a topology of 25 nodes and have shown a slightly better performance.

Sim and Sun (2003) [120]: `Multiple Ant Colony Optimization (MACO)`

In their work, the authors first presented an overview of ACO for routing and load balancing and then proposed the MACO approach for *load-balancing in connection-oriented networks*. MACO is based on the use of *multiple colonies*, where each colony lays its own type of pheromone. An ant is expected to select paths marked by high values of pheromone of the type laid by the colony the ant belongs to, and get repulsed by routes marked by high values of pheromone laid by ants of other colonies. This *anti-pheromone* mechanism is expected to be an efficient mechanism to find good *multiple disjoint paths*. The use of pheromone repulsion to favor the discovering of disjoint paths was earlier used by Navarro and Sinclair (1999) [91] to solve (static) problems of routing and wavelength allocation in all-optical networks.

Heegaard, Wittner, and Helvik (2003) [64]: `Cross-Entropy Ants (CE-Ants)`

CE-Ants shares the same forward-backward structure of AntNet but makes use of path updating formulae derived from Rubinstein's *Cross-Entropy* (CE) optimization framework [107]. The CE method is based on the repeated sampling of paths and on the consequent adaptive adjustment of γ, a parameter that biases path sampling, to minimize the cross-entropy between the used generation probabilities and the optimal importance sampling probabilities. In the distributed version of the CE algorithm designed by the authors, path sampling is implemented by the ants and is biased by the pheromone values. CE formulae are used to define how pheromone values are updated. The authors have also introduced the notion of *elitist ants*: only the best ants are allowed to trace back and update pheromone tables (see [24], Sect. 4.3.2, for a general discussion on the use and efficacy of elitist strategies in general ACO implementations). CE-ants has been applied to *virtual-path discovery*

and *failure management* in dynamic connection-oriented and label-switched IP networks offering some form of QoS. The authors have tested their approach considering the real backbone topology of Telenor, a major Norwegian network provider. In [63] Heegaard and Fuglem implemented and tested their system in a physical network using Linux routers.

5.6 Algorithms for Networks Providing Quality-of-Service

In this section we review the main work concerning the application of ACO ideas to wired networks providing QoS.

Di Caro and Vasilakos (2000) [39, 24]: ***AntNet and Stochastic Estimator Learning Automata (AntNet+SELA)***

AntNet+SELA is intended for *QoS routing in ATM networks*. Ant path sampling is complemented by the presence of *node agents* designed after *stochastic estimator learning automata* (SELA) [134, 89]. Each node agent exploits the information gathered by the ants to adaptively learn an effective routing policy for QoS traffic based on the use of a *link-state* routing table in addition to the usual ant pheromone table. Stochastic learning automata, have been used in early times [90, 92] to provide fully distributed adaptive routing. One of their main characteristics is that they learn by *induction*: no information is exchanged among the controllers. They only monitor local traffic and try to get an understanding of the effectiveness of the implemented routing choices. In AntNet+SELA, the static inductive learning component is enhanced by using the ants as active learners that gather also non-local information to keep up-to-date the link-state routing table to rapidly allocate resources for multipath QoS routing when requested.

In addition to the proactive ant generation as in AntNet-FA, at the arrival of a new session, the node manager reactively generate a *setup ant* and a group of *path probing ants*. The setup ant behaves similarly to the setup ants of AntNet-FS, with the difference that this time the ant searches for a path that strictly meets the QoS requirements. The path probing ants are *source routed*: each node agent uses its link-state database to compute the first k paths with minimum hop count that satisfy the QoS requests of the session, and assigns each one of these paths to a different probing ant that will check at run-time its availability and QoS consistency. According to the results provided by the backward ants, the node agent decides to whether or not accept the session and how to possibly split it over multiple paths. Unfortunately, the authors ran only few preliminary tests to evaluate the efficacy of the proposed model.

Oida and Sekido (1999) [95, 96]: ***Agent-based Routing System (ARS)***

ARS is an enhancement of AntNet that supports both best-effort and QoS routing based on an *IntServ model* with resource reservation and admission control. A Weighted Fair Queueing algorithm distributes at the nodes the

capacity between best-effort and QoS traffic. The QoS constraints considered are bandwidth and hop count. A real-time session can require one among n predefined levels of bandwidth and a number of hops less than a maximum value h. According to the basic AntNet scheme, from each node s ants are proactively generated and sent toward a sampled destination with the aim of finding a path with an available bandwidth that matches one of the n levels and with a hop count less than h. Links with more residual bandwidth are preferred to choose the next hop. If a feasible path is found, it is reported back to the source that stores it in a local cache which is kept up-to-date. When a real-time session requires a QoS path, the session is admitted or not according to the path information held in the cache. If a path that can meet the QoS requirements is present, an ant is sent to probe it and reserve the necessary resources. If the path is not there anymore, the session is rejected. Simulation results on a 14-nodes network show a high efficiency using network resources.

Michalareas and Sacks (2001) [85]: `Multi-Swarm`

The authors have exploited the main features of both AntNet and ABC to design an algorithm for routing in *multi-constrained* QoS networks. The algorithm provides soft QoS guarantees on end-to-end delay and bandwidth constraints, or, more in general, on additive (delay) and concave (bandwidth) constraints. Multi-Swarm deals with the two constraints adopting a *multi-colony* approach based on the use of two different swarm of ants, one for each constraint. The ants dealing with delay are in practice the same as in AntNet. On the other hand, since bandwidth is a non-additive metric and it cannot be directly measured from the ant, the authors have introduced a resource monitor that locally calculates the average spare bandwidth available at the links. When a bandwidth ant arrives at a node, it is artificially delayed for a time which is inversely proportional to the spare bandwidth, similarly to what happens in ABC. In this way, the bandwidth estimate is reduced to a delay estimate. Simulation experiments for three simple topologies under uniform TCP traffic shows that Multi-Swarm has performance comparable to OSPF.

Tadrus and Bai (2003) [127, 128, 126]: `QColony`

QColony is an algorithm for *QoS routing in multi-constrained networks* designed by extending and adapting AntNet behavior. QColony mostly addresses the *IntServ QoS model* but its structure makes it suitable to be used with other models such as DiffServ and MPLS. QColony categorizes network resources (e.g., bandwidth) in sets of adjacent *ranges*, where each range can fit a different QoS request from a user flow. For instance, if the resource is bandwidth, and, starting from the value of 0 Mbit/s, the network categorizes bandwidth requests in 10 ranges of 10 Mbit/s each, a user QoS request of 35 Mbit/s can be fit by all the seven upper ranges. At each node, learning and using good paths for each range is realized by associating to each range a unique vector of pheromone variables. In practice, this vector corresponds to the pheromone table normally used by AntNet-like algorithms to deal with

the case of best-effort traffic, which can be seen as a special case of QoS traffic with no traffic differentiation. Therefore, QColony, like Multi-Swarm, maintains *multi-pheromone tables*. This is reminiscent of what happens in nature, where different resources and events in the environment are dealt with different types of pheromone (see Sect. 4.1). In addition to QoS tables, a best-effort pheromone table is proactively maintained and used as in AntNet.

Upon receiving a QoS request, the ingress node determines the range which is suitable to satisfy the required QoS and reactively launches an *allocator ant* to find and reserve the resources. Allocator ants adopt a *greedy next hop selection* based on the pheromone values associated to the range they are looking for. If available, network resources are smartly allocated to accommodate the QoS request while at the same time leaving space for future requests. In addition to the allocator ants, QColony makes use of several other types of ants, all implementing greedy selections: (i) *explorer ants* are proactively generated and have a behavior analogous to AntNet ants, but on their backward journey they update pheromone entries associated to multiple ranges, (ii) *soldier ants*, mimicking the behavior of soldier ants in nature that respond to potentially harmful situations, are proactively generated to identify *short backup paths* to be used in case of failures along the paths in use by running flows, (iii) *maintenance ants* are reactively generated when a *path failure* happens, in this case they exploit the backup paths found by soldier ants to restore between the ingress and egress nodes the broken path. Using a custom simulator, the authors have made a number of simulation experiments to test QColony's performance versus the previously mentioned ARS, a probing-based reactive algorithm based on selective flooding [18], and QOSPF, which is a reference algorithm in the QoS domain. For small topologies and under low network traffic load the performance of the four algorithms is comparable, while QColony's performance is significantly better for large networks and heavy traffic loads.

Carrillo et al. (2004) [17, 16]: `AntNet-QoS`

AntNet-QoS is based on a *multi-pheromone* extension of AntNet to support QoS in a *DiffServ* network with m different classes of service for end-to-end delay. For each class, every node holds a pheromone table, a data routing table, and a vector of statistics, replicating in this way m times the data structures held by best-effort AntNet nodes. Ants are generated per class of service: they follow and update the pheromone table associated to their specific class. Ants are routed with higher priority than data, but respecting class-based queuing, such that the quality of their path reflects the class-specific conditions. Preliminary results are promising.

5.7 Algorithms for Wireless Mobile Ad Hoc Networks

In this section we review ant colony inspired algorithms for MANETs. Most of the implementations focus on the optimization of throughput and end-to-end delays. On the other hand, we will see that the bee-inspired algorithm

discussed later emphasizes battery optimization in addition to throughput and end-to-end delays.

Câmara and Loureiro (2000) [13, 14]: `GPS/Ant-Like algorithm (GPSAL)`

These authors were among the first to propose an ACO algorithm for MANETs. GPSAL is a *location-based* algorithm. It assumes and exploits the presence of an on-board GPS device. The routing information is exchanged locally among neighbors, and globally by sending forward ants to distant nodes addressed geographically. Ants are propagated through a bandwidth-efficient flooding algorithm. The algorithm achieves a similar performance with less routing overhead compared to LAR [75], another location-based algorithm.

Matsuo and Mori (2001) [82]: `Accelerated Ants Routing (AAR)`

AAR is based on the work of Subramanian et al. (see Subsect. 5.4). In AAR, uniform ants are equipped with a stack where the last n visited nodes are stored. This allows them to update the pheromone tables for all the last n intermediate nodes. The authors have compared AAR with AntNet, Q-routing and PQ-routing on a 56 node network and have shown its superior performance and faster convergence.

Guenes et al. (2002) [61]: `Ant-Colony-Based Routing Algorithm (ARA)`

ARA imports some basic aspects of AntNet into AODV. It is a *purely reactive* algorithm in which both forward and backward ants set up the paths to the nodes from which they arrive from. Also *data packets* update the pheromone tables reducing the number of ants needed to sample existing paths. According to simulation experiments, ARA's performance turns out to be slightly better compared to AODV but worse than DSR in highly dynamic environments.

Marwaha et al. (2002) [81]: `Ant-AODV`

In Ant-AODV, AODV is extended by a mechanism of proactive updating of the routing tables based on uniform ants. This increases the chance that a node or one of its neighbors will have a route to a destination when needed. The ants randomly traverse the network and keep track of the last n visited nodes. The results of simulation experiments indicate that Ant-AODV performs better than AODV and of a simple ant-based algorithm.

Baras and Mehta (2003) [4]: `Probabilistic Emergent Routing Algorithm (PERA)`

These authors have introduced two routing algorithms for MANETs. The first algorithm is a *proactive* one very similar to AntNet. Nodes maintain pheromone entries for all destinations by periodically launching forward ants, which take random decisions for unbiased exploration, and data packets are deterministically routed over the paths with the highest quality. The large routing overhead and the inefficient route discovery of this algorithm led to

PERA, which is purely *reactive* algorithm not very different from AODV. The forward ants are now flooded through the network towards their destinations. This strategy leads to the dynamic discovery of multiple paths. However, data packets are routed over the single best path available. The presence of multiple paths is helpful in the quick recovery from link failures. The performance of the algorithm is comparable to that of AODV according to a limited set of simulation experiments.

Heissenbüttel and Braun (2003) [65]: `Mobile Ant-Based Routing (MABR)`

The algorithm proposed by these authors makes use of geographical partitioning of the node area and of pheromone exploiting geographical addressing. The algorithm is intended for large-scale MANETs and is purely proactive. Forward/backward ants are used to periodically check if the path to a randomly chosen destination is functional and reflects the current state of the network. Accordingly, paths followed by the ants are positively or negatively reinforced. In addition, pheromone evaporation favors further exploration and removal of out-of-date paths.

Roth and Wicker (2003) [105, 104]: `Termite`

The Termite algorithm was actually inspired by the behavior of termite colonies, which is indeed very similar to that of ant colonies. As a matter of fact, Termite retains most of the main features of the general ACO metaheuristic such as pheromone tables, probabilistic decisions, pheromone evaporation, etc.. In Termite, forward ants are unicast and follow a random walk. Backward ants do not necessarily follow the forward path backward, but are also routed stochastically. Each data packet follows the path to its destination according to stochastic decisions based on the pheromone values, and "drops" pheromone indicating a path toward its source node. An exponential *pheromone evaporation* is introduced as a mean of negative feedback to prevent old routes from remaining in the routing tables. Termite is a *hybrid* algorithm. Paths are discovered on-demand by ants, but their goodness is implicitly sampled by data packets in a proactive fashion. The behavior and the properties of the algorithm have been studied using a *formal analysis* and by simulation, showing better performance than AODV.

Di Caro, Ducatelle, and Gambardella (2004) [33, 34, 51, 35, 36, 52]: `AntHocNet`

AntHocNet combines the typical *path sampling* behavior of ACO algorithms with a *pheromone bootstrapping* mechanism analogous to that used in Bellman-Ford algorithms (see Sect. 2), to effectively and efficiently learn pheromone tables. This design results in superior performance at the expenses of a relatively low routing overhead. AntHocNet is a *hybrid* algorithm. It is *reactive* in the sense that a node only starts gathering routing information for a specific destination when a local traffic session needs to communicate with the

destination and no routing information is available. It is *proactive* because as soon as the communication starts, and for the entire duration of the communication, the nodes proactively keep the routing information related to the ongoing flow up-to-date with network changes for both topology and traffic.

To capture the complexity of MANET environments, pheromone values reflect the quality of next hop decisions in terms of a *composite metric* function of: number of hops, traffic congestion, and signal-to-noise ratio (see [49] for a study on the effectiveness of considering different sets of quality metrics to define pheromone variables). This means that the algorithm tries to find paths characterized by minimal number of hops, low congestion, and good signal quality between adjacent nodes.

When a source node s starts a communication session with a destination node d, and no pheromone information is available about how to reach d, the node manager broadcasts a *reactive forward ant*. Ants are sent over high priority queues. At each node, the ant is either unicast or broadcast, according to whether or not the current node has pheromone information for d. If pheromone information is available, the ant stochastic decision policy π_ϵ makes use of a random proportional rule as in AntNet to select its next hop. Selection probabilities at node i are defined as: $p_{nd} = \frac{(\tau_{nd})^\beta}{\sum_{j \in \mathcal{N}_d^i} (\tau_{jd}^i)^\beta}$, where $\mathcal{N}_d^i$ is the set of i's neighbors over which a path to d is currently known, and $\beta \geq 1$ is a parameter which controls the exploratory behavior of the ants. A node which receives multiple copies of the same ant only accepts the first and discards the others. When a forward ants arrives at destination, it goes backward, updates the pheromone tables at the nodes, indicating a path between s and d, and triggers the sending of data packets from the traffic session. In this way, only one path is set up initially.

During the course of the communication session, additional paths are added and/or removed via a *proactive path maintenance and exploration* mechanism. This is implemented through a combination of ant path sampling and slow-rate *pheromone diffusion and bootstrapping* which mimics pheromone diffusion in nature. Each node n periodically and asynchronously broadcasts a sort of beacon message containing a list of destinations it has information about, including for each destination d its best pheromone value $\tau_{m^*d}^n$. A node i receiving the message from n, registers that n is its neighbor, and for each destination d listed in the message, it derives an estimate of the goodness of going from i to d over n combining the cost of hopping from i to n with the reported pheromone value $\tau_{m^*d}^n$. The authors call the obtained estimate b_{nd}^i *bootstrapped pheromone*, since it is built "bootstrapping" on the value of the path quality estimate received from an adjacent node

If i already has a pheromone entry τ_{nd}^i in its table, b_{nd}^i is just treated as an update of the goodness estimate of a known, reliable path, and is used directly to replace τ_{nd}^i with an up-to-date estimate. This equals to a *path maintenance* operation. If i does not have yet a value for τ_{nd}^i, b_{nd}^i *could* indicate a possible new path from i to d over n. However, this path has never been explicitly tried

out by an ant from i, such that due to the slow multi-step process it could have disappeared, or it can contain undetected loops or dangling links. The path is therefore not safe to use for data forwarding before being checked. This is the task assigned to *proactive forward ants*, which behave similarly to reactive forward ants but make use of both regular and bootstrapped pheromone on their way to the destination. This way, promising pheromone is checked out, and if the associated path is there and has the expected good quality, it can be turned into a regular path available for data. This *guided exploration* mechanism increases the number of paths available for data routing, which grows to a full mesh, and allows the algorithm to exploit new opportunities in the ever changing topology. *Stochastic decisions* are used to spread data packets over multiple paths with a strong preference for the best ones. *Link failures* are explicitly dealt with using a *local path repair* process that try to exploit the additional paths made available by the proactive mechanism, or via the generation of ant agents carrying explicit notification information.

AntHocNet's performance has been extensively evaluated through simulations against state-of-the-art algorithms under a number of different MANET scenarios for both *open space* [33, 34, 50, 51, 35, 2, 49] and realistic *urban* conditions [36, 52]. The authors studied the behavior of the algorithm under different conditions for network size (ranging from 50 to 1000 nodes), connectivity, change rate, data traffic patterns, and node mobility. The performance of the algorithm has been assessed relative to two classical MANET routing algorithms, AODV and OLSR, using the QualNet commercial simulator [101]. In the reported experiments, AntHocNet robustly outperforms the two competitor algorithms in terms of general efficacy, and in terms of adaptivity, robustness, and scalability. Superior performance were obtained efficiently introducing only a relatively small overhead, usually smaller than that introduced by the two other algorithms.

Rajagopalan and Shen (2005) [102, 103]: `Ad-Hoc Networking with Swarm Intelligence (ANSI)`

This work is based on earlier works of Shen [117, 118, 116]. ANSI is a *reactive* algorithm. Forward ants are reactively generated to look for a route for a new session or to repair a route after a link failure. They are *deterministically flooded* toward the destination. Only the first ant arriving at destination is converted to a source-routed backward ant that sets up the route. At each node i, the pheromone entry $\tau_{nd} \in \mathcal{T}^i$ represents a weighted measure of how many times the link $i \rightarrow n$ has been selected to go to d. Pheromone and routing tables are updated by both forward and backward ants, indicating and *reinforcing* the route for all nodes towards the starting node. Also the arrival of a data packet triggers an update, but only of the pheromone table. Contrary to what usually happens in ant algorithms, pheromone and routing tables are *not used for ant decisions*. Pheromone tables are used to derive *deterministic single-path* routing tables for data packets. At node i, the next hop r_d to be used for data bound for d is the next hop which has the highest value

$p_{nd} = \tau_{nd}^{\alpha}\eta_{nd}^{\beta}\psi_{nd},\ \forall n \in \mathcal{N}^i$, where η_{nd} is a heuristic measure of the inverse of the distance to d through n, ψ is an inverse heuristic measure of the congestion along the path, and α and β are appropriate weighting factors. Periodic sending of *Hello messages* is used to keep neighboring information up-to-date. The combination of Hello information and ant pheromone updates provides multiple paths for a destination, but only the best one is deterministically used to route data packets. *Pheromone evaporation* for all pheromone entries is triggered after each update to favor removal of unused and bad paths, which amounts to *negative reinforcement*. ANSI was shown to perform better than AODV in simulations experiments involving 50 mobile nodes.

6 Routing Protocols Inspired by Bee Colony Behaviors

Bee colony behaviors have driven the design of routing algorithms only more recently than ant colony behaviors. Most of the work in this sense has been done by Farooq and colleagues. They developed two main algorithms, `BeeHive`, for wired IP networks, and `BeeAdHoc`, for MANETs. From these two reference algorithms they have further derived other algorithms, addressing a number of different network constraints and scenarios. In the following we describe with some detail the characteristics of the two main reference algorithms and provide a brief discussion of also the additional algorithms and studies based on them. A peculiar characteristics of all this work consists in the fact that the algorithms have been designed according to the guidelines of so-called *natural engineering* framework [141]. That is, they have been designed keeping in mind the technical constraints of physical networks to avoid to make design assumptions which hardly would hold once the algorithm is being implemented on a real network. The efficacy of this way of proceeding is witnessed by the fact that the implementation of these algorithms in physical networks of Linux routers does not significantly differ at the algorithmic level from their implementation in network simulators, and, even more importantly, their performance in physical networks is consistent with what is observed in simulation experiments [53, 62].

6.1 The BeeHive Algorithm for Wired Connectionless Networks

The `BeeHive` algorithm [148, 141, 142, 53] is based on a meta routing framework similar to that of ACO-routing algorithms: paths are constantly tried out to discover new routes and adapt to changing network conditions, and data are spread over multiple paths to optimize network performance and resources utilization. This is achieved with a strategy that mimics bee foraging behaviors. More specifically, BeeHive is built around two types of agents, *short distance* and *long distance* bee agents, which are *proactively* generated at the nodes and are designed after the way bee foragers respond to waggle dances. Both types of agents undertake the same responsibility: *exploring* the

network and *evaluating* the quality of paths that they traverse to update node routing tables. However, short distance bee agents are allowed to move only up to a restricted number of hops. On the other hand, long distance bees have to collect and disseminate routing information in the complete network. This two-levels agent model is intended to quickly collect routing information while minimizing both processing and bandwidth overhead. BeeHive adopts a *hierarchical organization* of the network that matches the use of two different types of agents with different search ranges. The network is subdivided into *foraging zones* and *foraging regions*. A foraging zone is defined as the set of nodes around a given node from which short distance bee agents can reach the node. The same node may belong to foraging zones of many nodes. The network is also viewed as a collection of clusters of non-overlapping foraging regions, in which a node belongs to just one region only. Each foraging region has a *representative node*, which is the node with the lowest IP address in the region. Its role is to launch long distance bee agents. Each node maintains routing information for all nodes within its foraging zone, and for representative nodes of the foraging regions. If the destination of a packet does not lie within the foraging zone of a node, then it is forwarded along a path leading to the representative node of the foraging region containing the destination node. Informally, the behavior of BeeHive and its main characteristics can be summarized as follows:

1. All nodes start the *foraging region formation process* during a start-up phase. They try to form a foraging region with the same address as their own address and make themselves the representative node of the foraging region. They launch a *first generation of short distance bee agents* to propagate their identifier in their neighborhood.
2. If a node receives a short distance bee from a node whose representative node's address is smaller than that of the receiving node, then it discontinues its efforts to be a representative node and joins the foraging region of the node with the smaller address.
3. If a node later on learns that its representative node has joined another foraging region, then it repeats the region formation actions of Step 1.
4. Nodes keep on launching generations of short distance bee agents by following Steps 1–3 until the network is subdivided into *disjoint foraging regions*, and *overlapping foraging zones*. Finally, each node informs all other nodes in the network to which it belongs. This step is repeated every time foraging regions are reshaped because of link/node failures.
5. At the end of Step 4, the algorithm enters into a *normal phase* in which each non-representative node periodically sends out a short distance bee agent by *broadcasting* it to each one of its neighbors.
6. When a replica of a bee agent is received at a node, it updates the local routing information and is broadcast again to all the neighbors except to the one it was received from. This process continues until the lifetime of

the agent has expired, or the replica arrives at a node which has already received a copy of it.

7. Representative nodes only launch long distance bee agents that undergo the same process as the short distance ones but have longer lifetime.
8. Each node dynamically maintains routing information for reaching nodes within its foraging zone and for reaching the representative nodes of foraging regions. According to this hierarchical organization, a node routes a data packet whose destination is beyond its foraging zone along a path toward the representative node of the foraging region containing the destination node. More specifically, each node i maintains three types of routing tables: the *Intra Foraging Zone* (IFZ), the *Inter Foraging Region* (IFR), and the *Foraging Region Membership* (FRM) table. The *Intra Foraging Zone* routing table $\mathcal{R}^i$ is organized as a matrix of size $|\mathcal{D}^i| \cdot |\mathcal{N}^i|$, where $\mathcal{D}^i$ is the set of destinations in the foraging zone of node i and $\mathcal{N}^i$ is the set of neighbors of i. Each entry r_{jd} is a pair of queuing delay and propagation delay (q_{jd}, p_{jd}) that a packet will experience traveling to destination d via neighbor j. The Inter Foraging Region routing table stores the queuing and propagation delay values for reaching the representative node of each foraging region through each one of its neighbors. The Foraging Region Membership table provides the mapping of known destinations to foraging regions. Thanks to the hierarchical organization, the overall memory occupancy of routing information at each node is reduced with respect to the case of a flat organization, as the one adopted in many ACO implementations, that would require $O(|\mathcal{N}| \cdot |\mathcal{D}|)$ entries.
9. A bee agent launched from s, traveling across the network incrementally collects path information in the form of the *trip time estimate* t_{is} for reaching the source s from the current node i over the used link j. The main difference lies in the fact that path exploration by bee agents does not rely on a stochastic policy but is realized according to a *deterministic* scheme based on repeated broadcasting (i.e., *flooding*). Bee agents use *high priority queues* for quick dissemination of routing information.
10. The core mechanism at work in BeeHive is the direct agent-to-agent communication model (see Fig. 3) inspired by the bee behavior. In this topology, three paths exist between node k and node s. Node s launches three replicas of the same agent on three paths and they arrive at node k through different paths. Each replica uses the estimation model described above to estimate the queuing delay and the propagation delay. The replica that arrived earlier is allowed to continue its exploration further while other replicas are killed. However, the other replicas do communicate their estimates to the replica that is allowed to continue the exploration. Using the communication paradigm explained in Fig. 3 the replica calculates p_{ks} and q_{ks}, which incorporate the estimates of all replicas proportional to the quality of the paths, g_{1s}, g_{2s}, g_{3s} (to be shortly defined in Step 11), which they traversed. Once this replica continues its exploration of the network then it tells the other nodes that there exists a path from k to s through

which a packet could reach s with a propagation delay of p_{ks} and queuing delay of q_{ks}. The other nodes forward data packets to node k based on the quality which is a function of p_{ks} and q_{ks}. Once the data packet is at node k then it could take any one of the three paths based on their quality which is calculated based on the delay estimates of *bee agents*.

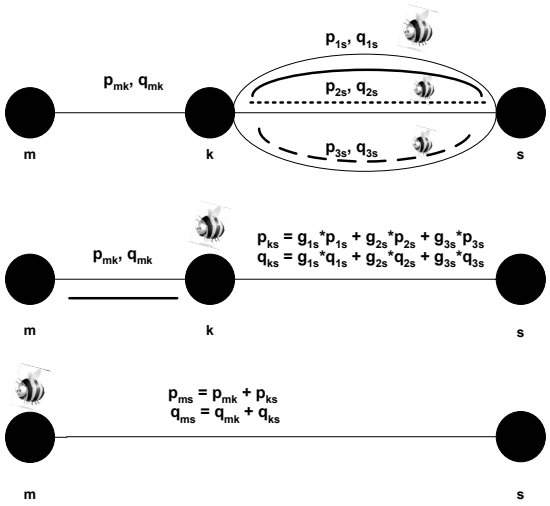

Fig. 3. Communication paradigm of *bee agents*

11. The next hop for a data packet is selected according to a *stochastic rule* that depends on the quality of each next hop. The quality of the path is a function of the cumulative queuing and propagation delays to reach the desired destination. The cumulative values are a result of the bee-inspired direct agent-to-agent communication model as depicted in Fig. 3. The estimated goodness of a neighbor j of node i (i has N neighbors) for reaching a destination d is g_{jd} and defined as

$$g_{jd} = \frac{\frac{1}{p_{jd}+q_{jd}}}{\sum_{k=1}^{N}(\frac{1}{p_{kd}+q_{kd}})}, \tag{7}$$

where p_{jd}, q_{jd} are respectively propagation and queuing delays estimated by the bee agents for reaching destination d via neighbor j of node i. The fundamental motivation behind this definition is to approximate the behavior of a real network. When the network is experiencing a heavy network traffic load, the queuing delay plays the primary role in the delay of a link. In this case it is trivial to say that $q_{jd} \gg p_{jd}$, and the goodness calculation becomes $g_{jd} \approx \frac{\frac{1}{q_{jd}}}{\sum_{k=1}^{N}\frac{1}{q_{kd}}}$. When the network is experiencing a

low traffic load, it is the propagation delay that plays an important role in defining the latency of a link. Since $q_{jd} \ll p_{jd}$, we obtain $g_{jd} \approx \frac{\frac{1}{p_{jd}}}{\sum_{k=1}^{N} \frac{1}{p_{kd}}}$.

Figure 4 provides an exemplary working of the flooding mechanism. Short distance bee agents can travel up to 2 hops in this example. Each replica of the shown bee agent (launched by Node 9) is specified with a different trail to identify its path unambiguously. The numbers on the paths show their cost. The flooding algorithm is a variant of the *breadth first* search algorithm. By following the above-mentioned Steps 1–4 the network is partitioned into two foraging regions with representative nodes 1 and 6 respectively. The foraging zone of Node 9, which spans over both foraging regions, consists of nodes 2–8. The bee agents utilize the following estimation model for the trip time t_{is} that a packet would experience to reach s from current node i coming from j (protocol processing delays are ignored):

$$t_{is} \approx \frac{l_{ij}}{b_{ij}} + tx_{ij} + d_{ij} + t_{js} \tag{8}$$

where l_{ij} is the size of the queue (in bits) for neighbor j at node i, b_{ij} is the bandwidth of the link between node i and neighbor j, such as $l_{ij}/b_{ij} = q_{ij}$, tx_{ij} and pd_{ij} are respectively transmission and propagation delay of the link between node i and neighbor j (i.e., $tx_{ij} + pd_{ij} = p_{ij}$), and t_{js} is the trip time from j to s. Bandwidth and propagation delays of all links of a node are calculated at the beginning by transmitting back and forth so-called *hello packets*. Bee agents have a fixed size of 48 bytes and currently they are launched after every second or when a node has received a certain number of packets (240 in the performed experiments).

BeeHive has been evaluated in considering an a testing framework designed to provide a robust evaluation of generic SI-based routing algorithms over a large set of operational scenarios [143, 53]. The authors have shown that with the help of this framework, they were able to discover some previously unknown behavior of the considered routing algorithms. The framework considers a number of auxiliary parameters that provide valuable insight into the performance vs. cost benefits of the routing protocols. The authors compared BeeHive, AntNet, AntNet-FA, DGA [77], and OSPF. Reported simulation results (obtained by custom implementations of the algorithms using the OMNeT++ simulator) showed that BeeHive was able to deliver same or better performance than AntNet, and solidly outperforms OSPF and DGA under heavy network traffic loads, while has a performance comparable to OSPF under low loads. An additional positive aspect of BeeHive is that it requires smaller routing tables and less computational resources than the considered competitors.

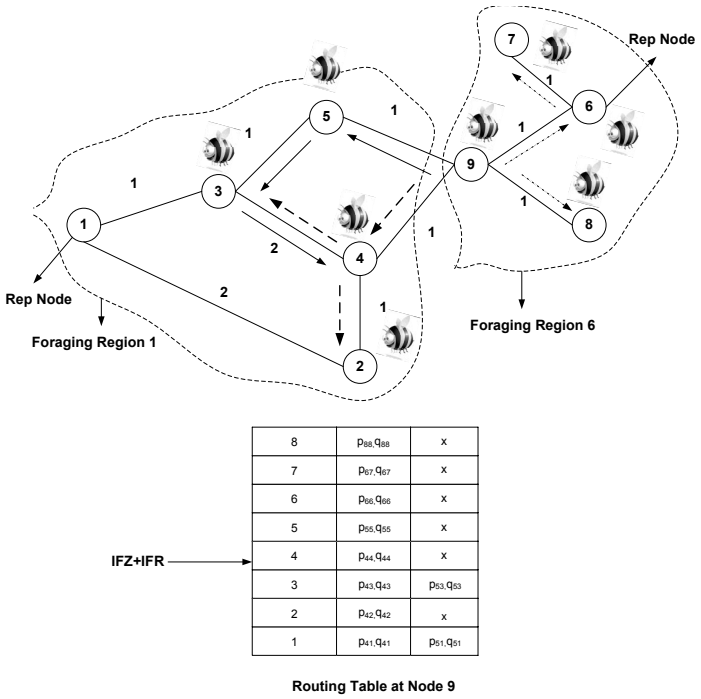

Fig. 4. Illustration of the working of the BeeHive algorithm.

Harsch, Wedde, and Farooq (2005,2006) [62, 53]: BeeHive implementation in Linux routers

BeeHive has been also tested in a *physical network*. In fact, the authors implemented BeeHive inside the network stack of Linux routers and then tested and compared its performance to OSPF in a relatively small network of Linux routers using different types of synthetic and real world traffic applications such as FTP and Voice over IP (VoIP) under UDP and TCP transport protocols. Results show that BeeHive can not only significantly help in reducing the download time of a file but also provide a better quality of service to VoIP sessions. Moreover, they have also shown that the performance of BeeHive in real-world networks is consistent with its performance in simulations. This work, confirming previously cited work on the physical implementation of ACO protocols, shows that an efficient and effective implementation of SI-based routing algorithms is possible, and it is highly competitive with current state-of-the-art algorithms.

6.2 Other Algorithms for Wired Networks Based on BeeHive

Wedde, Timm, and Farooq (2006) [150, 149]: `BeeHiveGuard, BeeHiveAIS`

The authors of these works have been the first to extensively analyze different types of security threats that malicious nodes can launch by manipulating the identity of agents or their routing information in networks controlled by SI-based algorithms. All these attacks can significantly degrade the network performance and compromise network operations. As a first step, they used a digital signature based security framework, BeeHiveGuard, to secure identity of a bee agent and of its routing information. The conclusion of the work was that BeeHiveGuard was able to counter different types of attacks but the processing complexity of the bee agents in BeeHiveGuard increased by more than 52000% and the control overhead increased by more than 200% as compared to BeeHive. Then, the authors proposed a novel solution, BeeHiveAIS, based on Artificial Immune System (AIS) ideas, that provided the same security level as the digital signature based framework but with processing and control overheads which are approximately 200 and 20 times respectively smaller than BeeHiveGuard. This is an important achievement toward the definition of a SI-based security protocol with manageable processing and communication costs.

Brüntrup and Farooq (2006) [12]: `BeeHivePlus, BeeHiveQoS`

In BeeHive data packets are spread stochastically over the available multiple paths according to their estimated quality. This is a behavioral trait common to many other algorithms reviewed in this chapter. While this way of proceeding has some clear benefits, on the other hand, in some situations requiring strict and predictable performance, such as in QoS networks, this can result in unwanted side-effects such as short-lived loops and jitter fluctuations. BeeHivePlus overcomes these potential problems by utilizing the concept of *waterfall routing*, in which only those neighbors are considered to be selected as a next hop that are nearer, in terms of hops, from the destination as compared to the existing node. As a result, a packet always move in the direction of the destination and loops are implicitly avoided. The authors have used the concept of *temporal stability* of routes i.e. routing decisions for a destination remain fixed between the inter-arrival time period of two successive bee agents from the same destination. This feature ensures that packets for a short time period follow the same path. Consequently, it results in significant reduction in jitter fluctuations that become comparable to that of a single-path algorithm such as OSPF. However, BeeHivePlus in spite of having these desirable features of loop freedom and low jitter, still provides similar performance as BeeHive. The same authors derived from BeeHivePlus a novel algorithm, BeeHiveQoS, for QoS networks. The core mechanism in BeeHiveQoS is an intelligent hierarchical packet scheduler, which can be embedded in BeeHivePlus as well as in other schemes, and which provides soft guarantees to QoS sensitive applications. The results of the experiments conducted

on network topologies up to 150 nodes confirm that BeeHiveQoS is able to provide guarantees to QoS sensitive applications.

Zahid, Shahzad, Ali, and Farooq (2007) [156]: Formal framework for performance analysis

The authors have proposed a formal framework for analyzing the behavior of the BeeHive protocol. The framework utilizes relevant concepts of deductive mathematics and queuing theory. The framework also uses Markov transition matrices and probabilistic recursive functions that significantly augment the formal understanding about different design options adopted in BeeHive. The authors have formally modeled the goodness of a neighbor that represents its quality to reach a destination, end-to-end packet delay, throughput and the probability of packets following loops with the help of their model. The authors have empirically validated the results obtained form their formal model with the ones obtained from OMNeT++ simulations on a small network topology. The estimated performance values of their formal model closely follow similar patterns as the values obtained through the network simulator. This work will be a cardinal step in removing a serious shortcoming of SI based algorithms: lack of formal understanding about their merits and behavior.

6.3 The BeeAdHoc Algorithm for Wireless Mobile Ad Hoc Networks

Wedde, Farooq, et al. (2004) [140, 144, 147, 146] designed `BeeAdHoc` with the aim of defining a MANET routing algorithm which is at the same time *energy-efficient* and provides performance comparable to those of existing state-of-the-art algorithms. Honey bee behavior served as the main inspiration to design the different types of agents at work in the system and their interaction. Adopting the bee metaphor, each node's routing controller is seen as an independent beehive where the bee agents live, act, and interact. BeeAdHoc is a relatively simple algorithm residing at the network layer that makes use of a *reactive* strategy for agent launching and of *source-routing* to forward packets. In the following we discuss in separate subsections the multi-agent model, the beehive-like architecture of the routers, and the performance of the algorithm.

Multi-agent model

BeeAdHoc is based on the use of four different bee-inspired types of agents: *packers, scouts, foragers*, and *beeswarms*.

Packers mimic the task of a food-storer bee. Packers reside inside a network node, receive and store data packets from the upper transport layer (see Fig. 5). Their main task is to find a forager for the data packet at hand. Once the forager is found and the packet is handed over, the packer agent is removed from the system.

The task of *scouts* is to discover new routes from their launching node to their destination node. A scout is broadcast to all neighbors in range using an *expanding time to live (TTL) timer* heuristic analogous to that used in the AODV algorithm [76]. At the start of the route search, a scout agent is generated, its TTL is set to a small value (e.g., 3), and is broadcast. If after a certain amount of time the scout is not back with a route, the strategy consists in the generation of a new scout and in the assignment of a TTL higher than in the previous attempt. In this way the search radius of the generated scouts is incrementally enlarged, increasing the probability of reaching the searched destination. When a scout reaches the destination, it starts a backward journey on the same route that it has followed while moving forward toward the destination. Once the scout is back to its source node, it recruits foragers for its route by utilizing a mechanisms derived from the waggle dance of scout bees in nature. A dance is abstracted into the number of clones that could be made of the same scout, which is encoded in their *dance number* (corresponding to recruiting forager bees in nature).

Foragers are the main workers in the BeeAdHoc algorithm. They are bound to the "bee hive" of a node. They receive data packets from packers and deliver them to their destination in a source-routed modality. To "attract" data packets foragers use the same metaphor of a waggle dance as scouts do. Foragers are of two types: delay and lifetime. From the nodes they visit, *delay foragers* gather end-to-end delay information, while *lifetime foragers* gather information about the remaining battery power. Delay foragers try to route packets along a minimum-delay path, while lifetime foragers try to route packets so that the lifetime of the network is maximized. A forager is transmitted from node to node using a unicast, point-to-point, modality. Once a forager reaches the searched destination and delivers the data packets, it waits there until it can be piggybacked on a packet bounded for its original source node. In particular, since TCP acknowledges received packets, BeeAdHoc piggybacks the returning foragers in the TCP acknowledgments. This reduces the overhead generated by control packets, saving at the same time energy.

Beeswarms are the agents that are used to explicitly transport foragers back to their source node when the applications are using an unreliable transport protocol like UDP, such that no acknowledgments are sent for the received data packets. To optimize forager transport, one beeswarm agent can carry multiple foragers: one forager is put in the header of the beeswarm while the others are put in the agent payload.

Beehive-like architecture of the node controllers

In BeeAdHoc, each MANET node contains at the network layer a software module called *hive*, which consists of three parts: the *packing floor*, the *entrance floor*, and the *dance floor*. The structure of the hive is shown in Fig. 5.

The entrance floor is an interface to the lower MAC layer, while the packing floor is an interface to the upper transport layer. The dance floor contains the

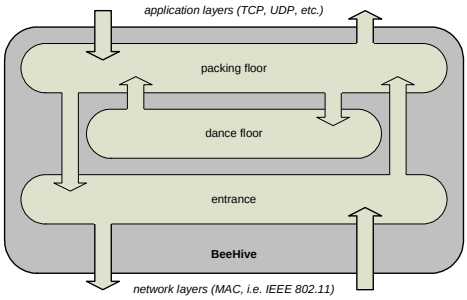

Fig. 5. Overview of the BeeAdHoc's *hive* architecture at a network node.

foragers and the routing information to route locally generated data packets. The functional characteristics of each floor composing the hive are as follows:

Packing floor. The packing floor is an interface to the upper transport layer (e.g., TCP or UDP). Once a data packet arrives from the transport layer, a matching forager for it is looked up on the dance floor. If one forager is found then the data packet is encapsulated in its payload. Otherwise, the data packet is temporary buffered waiting for a returning forager. If no forager comes back within a certain predefined time, a scout is launched which is responsible for discovering new routes to the packet destination.

Entrance floor. Actions at the dance floor depend on the type of packet that entered the floor from the MAC layer. If the packet is a forager and the current node is its destination, then the forager is forwarded to the packing floor, otherwise, it is directly routed to the MAC interface of the next hop node. If the packet is a scout agent, it is broadcast to the neighbor nodes if its TTL timer has not expired yet or if the current node is not its destination. The information about the ID of the scout and its source node is stored in a local list. If a replica of a previously received scout arrives at the entrance floor, the replica is removed from the system. If a forager with the same destination as the scout already exists in the dance floor, then the forager's route to the destination is given to the scout by appending it to the route held so far by the scout.

Dance floor. The dance floor is the heart of the hive because it maintains the *routing information* in the form of *foragers*. The dance floor is populated with routing information by means of a mechanism reminiscent of the waggle dance recruitment in natural bee hives: once a forager returns after its journey it recruits new foragers by "dancing" according to the quality of the path it traversed. A lifetime forager evaluates the quality of its route based on the average remaining battery capacity of the nodes along its route. Mimicking forager bees in nature that dance enthusiastically when they find a food source worth to be exploited, recruiting in this way a number of foragers, a lifetime forager can be cloned many times in two distinct cases. In the first case, the nodes on the discovered route have a

good amount of spare battery capacity, which means that this is a good route that can be well exploited. In the second case, a large number of data packets are waiting for the forager, so that the route needs to be exploited even though it might be having nodes with little battery capacity. On the other hand, if no data packets are waiting to be transported, then a forager with a very good route might even abstain from dancing because the other foragers are fully satisfying traffic requests. This concept is directly borrowed from the behavior of scout/forager bees in nature, and it helps to automatically regulate the number of foragers for each route.
The central activity of the dance floor module consists in sending a matching forager to the packing floor in response to a request from a packer. The foragers whose lifetime has expired are not considered for matching. If multiple foragers could be identified for matching then a forager is selected in a random way. This helps distributing the packets over *multiple paths*, which in turn, serves two purposes: avoiding congestion under high loads, and depleting batteries of different nodes at a comparable rate. A clone of the selected forager is sent to the packing floor and the original forager is stored in the dance floor after reducing its dance number, that is, the number of permitted clones. If the dance number is zero, then the original forager is sent to the packing floor, removing it in this way from the dance floor. This strategy aims at favoring young over old foragers, since they represent fresher routes, which are expected to remain valid in the near future with higher chances than the older ones due to mobility and battery depletion. If the last forager for a destination leaves a hive, then the hive does not have anymore a route to the destination. Nevertheless, if a route to the destination still exists, then soon a forager would be returning to the hive, while if no forager comes back within a reasonable amount of time then the node has probably lost its connection to the destination node. This mechanism eliminates the need for explicitly monitoring the validity of the routes by using special Hello packets and informing other nodes through route error messages, as it is done in several state-of-the-art algorithms such as AODV, as well as in several ACO implementations for MANETs. In this way less control packets are transmitted, resulting also in less energy expenditure.

Implementation and performance evaluation

BeeAdHoc has been implemented and evaluated both in simulation and in real networks. Results from extensive simulation tests show that BeeAdHoc delivers the same or better performance than that of the state-of-the-art algorithms like AODV, DSR and DSDV [98], but at a significantly smaller overall energy expenditure [140, 144, 146, 147]. To study its performance in more realistic and way more challenging physical networks, the authors have implemented BeeAdHoc inside the Linux network stack. They compared BeeAdHoc with

AODV and OLSR proposing a three step testing methodology with the intent of gradually moving towards a real MANET [140, 147]. First, in simulated reality, they tested the algorithms in a virtual network of 5 virtual machines connected through a software switch. They randomly changed the topology to simulate mobility. This scenario depicted an ideal MANET. Second, in quasi reality, the communication between laptops was established though 802.11 wireless network cards by placing laptops in the communication range of each other and the mobility was simulated by discarding packets through packet filtering at the link layer in case the laptop was not supposed to receive the packets. Finally, in real MANET, the authors conducted a MANET experiment on a 12 laptop network in which the nodes were moving at a walking speed at the north campus of University of Dortmund, Germany. Interestingly, performance values obtained in the simulated reality scenario did not show a correlation with those of the real MANET. However, the overall pattern of the values remained the same: BeeAdHoc was able to deliver consistent and similar/better performance as compared to OLSR and AODV but at the cost of significantly less control packets and consuming less battery power.

Schletter, Fischer, et al. (2005) [147]: An agent-based formal investigation of BeeAdHoc

The formal analysis of routing protocols for MANETs is a challenging area of research. In [147] the authors have studied performance and behavior of BeeAdHoc adopting as a *formal verification model*, which is a rather innovative approach in the domain of SI. They relied on the framework of Sumpter [124] to propose an agent interaction model based on the use of the Weight Synchronous Calculus of Communicating Systems (WSCCS). Using a probabilistic workbench to validate the formal model, the authors have shown that the formal model is able to predict the distribution of the foragers on multiple paths as a function of their quality. Moreover, they were also able to analyze the sending pattern of beeswarms and optimize it through the model. This work can play a vital role in developing a comprehensive formal model checking framework for SI based routing algorithms in general.

6.4 Other Algorithms for MANETs Based on BeeAdHoc

Mazhar and Farooq (2007) [83, 84]: `BeeSec, BeeAIS`

The authors followed the same research methodology as was used in BeeHiveGuard and BeeHiveAIS to analyze the security threats of BeeAdHoc and then proposed two solutions: BeeSec, which utilizes a digital signature based security framework and BeeAIS, which utilizes the principles of AIS to provide security. However, in MANETS, providing an AIS based security is more challenging because of the mobility of the nodes. As a result, it was difficult to identify whether the change in the path of an agent is due to the malicious activity of a node or to its internal mobility. This translates into

the idea of a "self" which is changing. According to the reported results: (i) BeeAIS provides the same security level as compared to BeeSec but at significantly smaller processing and communication costs, which means a significant amount of energy and power saving compared to BeeSec; (ii) the performance of BeeAIS, even with the overhead for providing security, is significantly better as compared to AODV and DSR and is approximately similar to that of the original BeeAdHoc algorithm. These results seem to indicate the efficacy of adopting an AIS component for SI-based security protocols in power-aware systems, considering the low processing complexity and the absence of additional communication costs.

7 Conclusions and future perspectives for SI routing

In this chapter we addressed the problem of *routing* in current and next generation telecommunications networks, which are characterized by the fact of being very complex, dynamic, large, and heterogeneous. Swarm intelligence design features a number of properties that are highly desirable to deal with the challenges posed by these networks. In the chapter we reviewed the major routing protocols inspired by collective behaviors observed in *social insects* such as *ant* and *bee colonies*. This specific class of *SI algorithms* includes the majority of the most significant and promising applications of the SI paradigm to the solutions of adaptive network routing problems.

Social insects behaviors, mainly ant colonies, have fueled in the last 10 years the design and implementation of a consistent number of protocols and algorithms for routing. We could not review here this whole body of work, but we provided anyway a quite comprehensive overview of the main algorithms, of their general design principles and general properties. More specifically, we discussed the *pheromone-driven shortest-path behavior* of foraging ant colonies, which has been reverse-engineered and put to work in the optimization framework of *ant colony optimization*, that, in turn, has guided the design of a relatively large number of routing algorithms. Analogously, we pointed out and abstracted those core mechanisms at work in foraging bee colonies, such as waggle dancing, which have recently driven the design of novel routing algorithms. We have considered different classes of networks characterized by different communication and transmission technology and provided services, and for each class we have briefly discussed the characteristics of the main ant- and bee-colony inspired algorithms which can be found in literature. We have pointed out their pros and cons and their distinctive features in relationship to the classification features of routing algorithms given in Sect. 3. Tables 1 and 2 summarize these core features for a few of the most prominent among the reviewed algorithms, and compare them to established/classical state-of-the-art routing algorithms. From the tables it is apparent that, in general, SI algorithms have and make use of certain features that classical algorithms do not have and vice versa. On the other hand, since according to extensive

comparative studies the performance of the reviewed algorithms, and in particular of those listed in the tables, seems to be significantly better than that of classical state-of-the-art algorithms, one might argue that the properties implied by the SI design are particularly suitable to face the challenges of modern networks.

Generally speaking, a fundamental aspect of the SI paradigm is the fact that it emphasizes a particular *bottom-up design approach* which, for network routing results in the definition of protocols featuring among others: *locality of interactions and self-organizing behaviors, availability of multiple paths for routing and failure backup, ability to adapt in a quick and robust way to topological and traffic changes and component failures, scalable performance, robustness to failures and losses internal to the protocol, easiness of design and tuning.* As it is also confirmed by the data in the tables, most of the reviewed algorithms possess significant subsets of these important properties which are particularly appealing for current and future networking. In fact, with the impressive growth of the Internet, and the pervasive deployment of wireless and wired networking, these are the core properties which should characterize all modern protocols of the network protocol stack to be able to cope with the levels of complexity, heterogeneity, and dynamism of current and forthcoming networks. The relatively novel fields of *traffic engineering* and *autonomic communications* precisely emphasize the need for both an efficient utilization of network resources and the ability of the network to self-control and adapt over time both as a whole and at the level of the single components. "Classical" algorithms for network management and control have been designed top-down not taking into account the explosion in complexity in all directions and dimensions faced by current networks. Again, this can be also understood looking at the tables. It is clear that established classical algorithms miss some core properties in terms of dynamic behavior, robustness, and locality.

We believe that routing algorithms inspired by social insects behaviors, and, more generally, by the SI paradigm, can play an important role to empower future networks with an optimized, adaptive, robust, and scalable control system at the network layer. The downside of these novel approaches consists in the *current lack of extensive implementation and testing on physical networks*, and in the difficulty, somehow intrinsic to fully distributed and stochastic bottom-up approaches, to provide formal guarantees in terms of *dependability.* Solid work in these two aspects is still necessary to get a wider acceptance from the networking community and a deeper and solid understanding of the behavior and properties of these algorithms. If this will be done in the near future, we can expect a rapid deployment of SI algorithms in the control system of fortcoming networks.

Table 1. General features of routing algorithms for wired networks. The considered features have been discussed in Sect. 3. The algorithms have been discussed in the previous sections. "y" and "n" stand respectively for "yes" and "no", while "p" means that the algorithm partly possesses the feature

.

	AntNet	**Adaptive-SDR**	**BeeHive**	**OSPF**	**MDVA**	**QColony**	**QOSPF**
Topology-adaptive	p	y	y	y	y	p	y
Traffic-adaptive	y	y	y	n	p	y	y
Router-Intelligent	y	y	y	y	y	y	n
Multi-path	y	y	y	n	y	n	n
Local representation	y	y	y	n	n	y	n
Hierarchical	n	y	y	y	n	y	y
Constructive	n	n	y	y	n	n	y
Loop-free	n	n	p	y	y	n	y
Proactive behavior	y	y	y	y	y	y	y
Reactive behavior	n	n	n	n	n	y	n
Stochastic exploration	y	y	n	n	n	p	n
Stochastic data routing	y	y	y	n	n	n	n
Formal properties	n	n	y	y	y	n	p
Physical implementation	y	n	y	y	n	n	y
Quality of service	n	n	n	n	n	y	y

Table 2. General features of routing algorithms for MANETs. The considered features have been discussed in Sect. 3. The algorithms are discussed in the previous sections. "y" and "n" stand respectively for "yes" and "no", while "p" means that the algorithm partly possesses the feature.

	AntHocNet	**ANSI**	**Termite**	**BeeAdHoc**	**DSR**	**AODV**	**OLSR**
Topology-adaptive	y	y	y	y	y	y	y
Traffic-adaptive	y	y	y	y	p	p	p
Router-Intelligent	y	y	y	n	n	y	y
Multi-path	y	p	y	y	n	n	n
Local representation	y	y	y	y	y	y	n
Hierarchical	n	n	n	n	n	n	n
Constructive	y	y	y	y	y	y	n
Loop-free	n	n	n	y	y	n	n
Proactive behavior	y	p	y	n	n	n	y
Reactive behavior	y	y	y	y	y	y	n
Stochastic exploration	y	n	y	n	n	n	n
Stochastic data routing	y	n	y	y	n	n	n
Formal properties	n	n	p	p	p	p	p
Physical implementation	p	n	n	y	y	y	y
Energy-aware	n	n	n	y	n	n	n

References

1. S. Appleby and S. Steward. Mobile software agents for control in telecommunications networks. *BT Technology Journal*, 18(1):68–70, 2000.
2. O. Babaoglu, G. Canright, A. Deutsch, G. A. Di Caro, F. Ducatelle, L. M. Gambardella, N. Ganguly, M. Jelasity, R. Montemanni, A. Montresor, and T. Urnes. Design patterns from biology for distributed computing. *ACM Transactions on Autonomous and Adaptive Systems*, 1(1):26–66, 2006.
3. B. Baran and R. Sosa. A new approach for AntNet routing. In *Proceedings of ICCCN*, pages 303–308, Las Vegas, NV, USA, 2000. IEEE Press.
4. J. S. Baras and H. Mehta. A probabilistic emergent routing algorithm (PERA) for mobile ad hoc networks. In *Proceedings of WiOpt*, 2003.
5. D. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, USA, 1995.
6. D. Bertsekas and R. Gallager. *Data Networks*. Prentice–Hall, Englewood Cliffs, NJ, USA, 1992.
7. E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, Inc., New York, NY, USA, 1999.
8. E. Bonabeau, F. Henaux, S. Guérin, D. Snyers, P. Kuntz, and G. Theraulaz. Routing in telecommunications networks with ant-like agents. In *Proceedings of IATA*, pages 60–71, London, UK, 1998. Springer-Verlag.
9. J.A. Boyan and M.L. Littman. Packet routing in dinamically changing networks: A reinforcement learning approach. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Proceedings of NIPS6*, pages 671–678. Morgan Kaufmann, San Francisco, CA, USA, 1994.
10. R. W. Brazier and M. D. Cookson. Intelligence design patterns. *BT Technology Journal*, 23(1):69–81, 2005.
11. J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of MobiCom*, pages 85–97, New York, NY, USA, 1998. ACM Press.
12. R. Brüntrup. Quality of service in von der natur inspirierten routing-algorithmen (in german). Master thesis, LSIII, The University of Dortmund, Germany, August 2006.
13. D. Câmara and A.F. Loureiro. A novel routing algorithm for ad hoc networks. In *Proceedings of HICSS*. IEEE Press, 2000.
14. D. Câmara and A.F. Loureiro. GPS/Ant-like routing in ad hoc networks. *Telecommunication Systems*, 18(1–3):85–100, 2001.
15. T. Camilo, C. Carreto, J. Sá Silva, and F. Boavida. An energy-efficient ant-based routing algorithm for wireless sensor networks. In *Proceedings ANTS*, volume 4150 of *Lecture Notes in Computer Science*, pages 49–59, Brussels, Belgium, 2006. Springer.
16. L. Carrillo, C. Guadal, J.-L. Marzo, G.A. Di Caro, F. Ducatelle, and L.M. Gambardella. Differentiated quality of service scheme based on the use of multiple classes of ant-like mobile agents. In *Proceedings of CoNEXT*, pages 234–235, Toulouse, France, October 24–27 2005. ACM Press.
17. L. Carrillo, J. L. Marzo, L. Fàbrega, P. Vilà, and C. Guadall. Ant colony behaviour as routing mechanism to provide quality of service. In *Proceedings of ANTS*, volume 3172 of *Lecture Notes in Computer Science*, pages 418–419, Berlin, 2004. Springer.

18. S. Chen and K. Nahrstedt. An overview of quality-of-service routing for the next generation high-speed networks: Problems and solutions. *IEEE Network Magazine, Special issue on Transmission and Distribution of Digital Video*, 12(6):64–79, 1998.
19. S.P. Choi and D.-Y. Yeung. Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control. In *Proceedings of NIPS8*, pages 945–951. MIT Press, 1996.
20. Cisco. Internetworking technology handbook, 2002.
21. T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *Proceedings of IEEE INMIC*, pages 62– 68. IEEE Press, 2001.
22. M.E. Csete and J.C. Doyle. Reverse engineering of biological complexity. *Science*, 295(5560):1664–1669, March 2002.
23. S.S. Dhillon and P. Van Mieghem. Performance analysis of the AntNet algorithm. *Computer Networks*, 51(8):2104–2125, 2007.
24. G.A. Di Caro. *Ant Colony Optimization and its application to adaptive routing in telecommunication networks.* PhD thesis, Faculté des Sciences Appliquées, Université Libre de Bruxelles, Brussels, Belgium, November 2004.
25. G.A. Di Caro and M. Dorigo. Adaptive learning of routing tables in communication networks. In *Proceedings of the Italian Workshop on Machine Learning (IWML)*, Torino, Italy, December 9-10 1997.
26. G.A. Di Caro and M. Dorigo. AntNet: A mobile agents approach to adaptive routing. Technical Report 97–12, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, June 1997.
27. G.A. Di Caro and M. Dorigo. Ant colonies for adaptive routing in packet-switched communications networks. Technical Report 97-20.1, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, March 1997.
28. G.A. Di Caro and M. Dorigo. Mobile agents for adaptive routing. In *Proceedings of HICSS*, volume 7, pages 74–83. IEEE Computer Society Press, 1998.
29. G.A. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research (JAIR)*, 9:317–365, 1998.
30. G.A. Di Caro and M. Dorigo. Ant colonies for adaptive routing in packet-switched communications networks. In *Proceedings of PPSN-V*, volume 1498 of *LNCS*, pages 673–682. Springer, 1998.
31. G.A. Di Caro and M. Dorigo. Extending AntNet for best-effort Quality-of-Service routing. Proceedings of the First Internation Workshop on Ant Colony Optimization (ANTS'98), 1998.
32. G.A. Di Caro and M. Dorigo. Two ant colony algorithms for best-effort routing in datagram networks. In *Proceedings of the 11th International Conference on Parallel and Distributed Computing Systems (PDCS)*, pages 541–546, 1998.
33. G.A. Di Caro, F. Ducatelle, and L.M. Gambardella. AntHocNet: an ant-based hybrid routing algorithm for mobile ad hoc networks. In *Proceedings of PPSN-VIII*, volume 3242 of *LNCS*, pages 461–470. Springer, 2004. (Best paper award).
34. G.A. Di Caro, F. Ducatelle, and L.M. Gambardella. AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications*, 16(5):443–455, 2005.
35. G.A. Di Caro, F. Ducatelle, and L.M. Gambardella. Swarm intelligence for routing in mobile ad hoc networks. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 76–83, Pasadena, USA, June 2005. IEEE Press.

36. G.A. Di Caro, F. Ducatelle, and L.M. Gambardella. Studies of routing performance in a city-like testbed for mobile ad hoc networks. Technical Report 07-06, IDSIA, Lugano, Switzerland, March 2006.
37. G.A. Di Caro, F. Ducatelle, and L.M. Gambardella. Swarm intelligence for routing in telecommunications networks. *Journal of Swarm Intelligence*, 2007. Submitted.
38. G.A. Di Caro, F. Ducatelle, and L.M. Gambardella. Theory and practice of Ant Colony Optimization for routing in dynamic telecommunications networks. In N. Sala and F. Orsucci, editors, *Reflecting interfaces: the complex coevolution of information technology ecosystems*. Idea Group, Hershey, PA, USA, 2008.
39. G.A. Di Caro and T. Vasilakos. Ant-SELA: Ant-agents and stochastic automata learn adaptive routing tables for QoS routing in ATM networks. In Proceedings of 2nd International Workshop on Ant Colony Optimization (ANTS'00), 2000.
40. S. Doi and M. Yamamura. BntNetL: Evaluation of its performance under congestion. *Journal of IEICE B* (in Japanese), pages 1702–1711, 2000.
41. S. Doi and M. Yamamura. BntNetL and its evaluation on a situation of congestion. *Electronics and Communications in Japan (Part I)*, 85(9):31–41, 2002.
42. S. Doi and M. Yamamura. An experimental analysis of loop-free algorithms for scale-free networks. In *Proceedings of ANTS*, volume 3172 of *Lecture Notes in Computer Science*, pages 278–285. Springer-Verlag, 2004.
43. M. Dorigo, E. Bonabeau, and G. Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8):851–871, 2000.
44. M. Dorigo and G.A. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, 1999.
45. M. Dorigo, G.A. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.
46. M. Dorigo and E. Sahin (Editors). Special issue on Swarm Robotics. *Autonomous Robots*, 17(2–3), 2004.
47. M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 26(1):29–41, 1996.
48. M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.
49. F. Ducatelle, G.A. Di Caro, and L. M. Gambardella. An analysis of the different components of the AntHocNet routing algorithm. In *Proceedings of the 5th International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS'06)*, volume 4150 of *LNCS*, pages 37–48. Springer, 2006.
50. F. Ducatelle, G.A. Di Caro, and L.M. Gambardella. Ant agents for hybrid multipath routing in mobile ad hoc networks. In *Proceedings of WONS*, Switzerland, January 18–19, 2005. IEEE Press.
51. F. Ducatelle, G.A. Di Caro, and L.M. Gambardella. Using ant agents to combine reactive and proactive strategies for routing in mobile ad hoc networks. *International Journal of Computational Intelligence and Applications,* Special Issue on Nature-Inspired Approaches to Networks and Telecommunications, 5(2):169–184, 2005.
52. F. Ducatelle, G.A. Di Caro, and L.M. Gambardella. An evaluation of two swarm intelligence manet routing algorithms in an urban environment. In

Proceedings of the 5th IEEE Swarm Intelligence Symposium, St. Louis, USA, September 21–23, 2008.
53. M. Farooq. *Intelligent Network Traffic Engineering through Bee-inspired Natural Protocol Engineering.* Natural Computing Series. Springer, (In Press).
54. L. M. Feeney. A taxonomy for routing protocols in mobile ad hoc networks. Technical Report ISRN:SICS-T-99/07-SE, Swedish Institute of Computer Science, Kista, Schweden, 1999.
55. S. Fenet and S. Hassas. An ant based system for dynamic multiple criteria balancing. Proceedings of the First Internation Workshop on Ant Colony Optimization (ANTS'98), 1998.
56. S. Fenet and S. Hassas. A.N.T.: a distributed network control framework based on mobile agents. In *Proceedings of the International ICSC Congress on Intelligent Systems and Applications*, 2000.
57. R. Freeman. *Telecommunication System Engineering.* John Wiley & Sons, 2004.
58. M. Gadomska and A. Pacut. Performance of ant routing algorithms when using TCP. In M. Giacobini *et al.*, editor, *Applications of Evolutionary Computing, EvoWorkshops2007*, volume 4448 of *Lecture Notes in Computer Science*, pages 1–10. Springer Verlag, 2007.
59. R.M. Garlick and R.S. Barr. Dynamic wavelength routing in WDM networks via Ant Colony Optimization. In M. Dorigo, G.A. Di Caro, and M. Sampels, editors, *Proceedings of ANTS*, volume 2463 of *Lecture Notes in Computer Science*, pages 250–255. Springer Verlag, 2002.
60. S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 76:579–581, 1989.
61. M. Günes, U. Sorges, and I. Bouazizi. ARA - the ant-colony based routing algorithm for MANETS. In *Proceedings of the 2002 ICPP International Workshop on Ad Hoc Networks (IWAHN 2002)*, pages 79–85, 2002.
62. A. Harsch. Design and development of a network infrastructure for swarm routing protocols inside linux. Master thesis, LSIII, The University of Dortmund, Germany, July 2005.
63. P. Heegaard and I. Fuglem. AntPing: prototype demonstrator of swarm based path management and monitoring (Poster). In *Proceedings of IWSOS*, 2006.
64. P. Heegaard, O. Wittner, and B. Helvik. Self-management of virtual paths in dynamic networks. In Ozalp Babaoglu, Márk Jelasity, Alberto Montresor, Christof Fetzer, Stefano Leonardi, Aad van Moorsel, and Maarten van Steen, editors, *Self-Star Properties in Complex Information Systems*, volume 3460 of *Lecture Notes in Computer Science*, pages 417–432. Springer-Verlag, 2005.
65. M. Heissenbüttel and T. Braun. Ants-based routing in large scale mobile ad-hoc networks. In *13. ITG/GI-Fachtagung Kommunikation in verteilten Systemen (KiVS 2003)*, pages 91–99, Leipzig, Germany, 2003.
66. M. Heusse, D. Snyers, S. Guérin, and P. Kuntz. Adaptive agent-driven routing and load balancing in communication networks. *Advances in Complex Systems*, 1(2):237–254, 1998.
67. M. Heusse, D. Snyers, and Y. Kermarrec. Adaptive agent driven routing in communication networks: comparison with a classical approach. *Advances in Complex Systems*, 2(3):209–219, 1999.
68. N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, 21(6):879–894, 2003.

69. D. E. Jackson and F. L. Ratnieks. Communication in ants. *Current biology*, 16(15):570–574, 2006.
70. D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.
71. I. Kassabalidis, M. A. El-Sharkawi, R. J. Marks, P. Arabshahi, and A. A. Gray. Swarm intelligence for routing in communication networks. In *Proceedings of GLOBECOM*, pages 3613–3617. IEEE, IEEE Press, 2001.
72. I. Kassabalidis, M.A. El-Sharkawi, R.J. Marks II, P. Arabshahi, and A.A. Gray. Adaptive-SDR: Adaptive swarm-based distributed routing. In *Proceedings of IJCNN*, pages 351–354. IEEE Press, 2002.
73. J. Kephart and D. Chess. The vision of autonomic computing. *IEEE Computer magazine*, 36(1):41–50, January 2003.
74. A. Khanna and J. Zinky. The revised ARPANET routing metric. *ACM SIGCOMM Computer Communication Review*, 19(4):45–56, 1989.
75. Y.-B Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile Ad Hoc networks. In *Proceedings of MOBICOM*, pages 66–75. ACM Press, 1998.
76. S.-J. Lee, E. M. Royer, and C. E. Perkins. Scalability study of the ad hoc on-demand distance vector routing protocol. *ACM/Wiley International Journal of Network Management*, 13(2):97–114, March 2003.
77. S. Liang, A. N. Zincir-Heywood, and M. I. Heywood. The effect of routing under local information using a social insect metaphor. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, May 2002.
78. S. Liang, A. N. Zincir-Heywood, and Malcolm I. Heywood. Adding more intelligence to the network routing problem: AntNet and GA-agents. *Applied Soft Computing*, 6(3):244–257, 2006.
79. S. Liang, A.N. Zincir-Heywood, and M.I. Heywood. Intelligent packets for dynamic network routing using distributed genetic algorithm. In *Proceedings of GECCO*, pages 88–96. ACM Press, 2002.
80. G. S. Malkin. *RIP: An Intra-Domain Routing Protocol*. Addison-Wesley, 1999.
81. S. Marwaha, C. K. Tham, and D. Srinavasan. Mobile Agents based Routing Protocol for Mobile Ad hoc Networks. In *Proceedings of GLOBECOM*, pages 163–167, Taipei, Taiwan, November 2002. IEEE Press.
82. H. Matsuo and K. Mori. Accelerated ants routing in dynamic networks. In *Proceedings of SNPD*, pages 333–339, August 2001.
83. N. Mazhar and M. Farooq. BeeAIS: Artificial immune system security for nature inspired, MANET routing protocol, BeeAdHoc. In *Proceedings of the 6th International Conference on Artificial Immune Systems*, volume 4628 of *LNCS*. Springer, 2007.
84. N. Mazhar and M. Farooq. Vulnerability analysis and secuirty framework (BeeSec) for nature inspired manet routing protocols. In *Proceedings of GECCO*, pages 102–109. ACM Press, 2007.
85. T. Michalareas and L. Sacks. Link-state and ant-like algorithm behaviour for single-constrained routing. Proceedings of the IEEE Workshop on High Performance Switching and Routing, May 2001.
86. T. Michalareas and L. Sacks. Stigmergic techniques for solving multi-constraint routing for packet networks. In *Proceedings of the First International Conference on Networking (ICN), Part II*, volume 2094 of *Lecture Notes in Computer Science*, pages 687–697. Springer-Verlag, 2001.

87. J. Moy. *OSPF: Anatomy of an Internet Routing Protocol.*Addison-Wesley,1998.
88. R. Muraleedharan and L.A. Osadciw. A predictive sensor network using ant system. In R.M. Rao, S.A. Dianat, and M.D. Zoltowski, editors, *Digital Wireless Communications VI, Proceedings of the SPIE*, pages 181–192, 2004.
89. K. Narendra and M Thathachar. *Learning Automata: An Introduction.* Prentice-Hall, 1989.
90. K. S. Narendra and M. A. Thathachar. On the behavior of a learning automaton in a changing environment with application to telephone traffic routing. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-10(5):262–269, 1980.
91. G. Navarro-Varela and M. C. Sinclair. Ant colony optimisation for virtual-wavelength-path routing and wavelength allocation. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 1809–1816, 1999.
92. O. V. Nedzelnitsky and K. S. Narendra. Nonstationary models of learning automata routing in data communication networks. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17:1004–1015, 1987.
93. The NS-2 network simulator. `http://nsnam.isi.edu/nsnam/`.
94. K. Oida and A. Kataoka. Lock-free AntNet and its evaluation adaptiveness. *Journal of IEICE B (in Japenese)*, J82-B(7):1309–1319, 1999.
95. K. Oida and M. Sekido. An agent-based routing system for QoS guarantees. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 833–838, 1999.
96. K. Oida and M. Sekido. ARS: An efficient agent-based routing system for QoS guarantees. *Computer Communications*, 23:1437–1447, 2002.
97. S. Okdem and D. Karaboga. Routing in wireless sensor networks using Ant Colony Optimization. In *Proceedings of AHS*, pages 401–404, 2006.
98. C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of SIGCOMM*, pages 234–244. ACM Press, 1994.
99. C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of WMCSA*, pages 90–100. IEEE Press, 1999.
100. L. Peshkin, N. Meuleau, and L.P. Kaelbling. Learning policies with external memory. In *Proceedings of ICML*, pages 307–314, 1999.
101. Qualnet Simulator, Version 3.9. Scalable Network Technologies, Inc., Culver City, CA, USA, 2005. `http://www.scalable-networks.com`.
102. S. Rajagopalan and C. Shen. ANSI: A unicast routing protocol for mobile ad hoc networks using swarm intelligence. In *Proceedings of the International Conference on Artificial Intelligence (ICAI)*, pages 24–27, 2005.
103. S. Rajagopalan and C.-C Shen. ANSI: a swarm intelligence-based unicast routing protocol for hybrid ad hoc networks. *Journal of System Architecture*, 52(8-9):485–504, 2006.
104. M. Roth and S. Wicker. Termite: Ad-hoc networking with stigmergy. In *Proceedings of IEEE GLOBECOM*, pages 2937–2941, 2003.
105. M. Roth and S. Wicker. Termite: Emergent ad-hoc networking. In *Proceedings of the 2nd Mediterranean Workshop on Ad-Hoc Networks (Med-Hoc-Net)*, 2003.
106. E. M. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 6(2):46–55, 1999.
107. R. Y. Rubinstein. Combinatorial optimization, cross-entropy, ants and rare events. In S. Uryasev and P.M. Pardalos, editors, *Stochastic Optimization: Algorithms and Applications*, pages 304–358. Kluwer Academic Publisher, 2000.

108. M. Saleem and M. Farooq. Beesensor: A bee-inspired power aware routing protocol for wireless sensor networks. In *In M. Giacobini et al. (Eds.), Lecture Notes in Computer Science, LNCS 4449*, pages 81–90. Springer Verlag, 2007.
109. M. Saleem and M. Farooq. A framework for empirical evaluation of nature inspired routing protocols for wireless sensor networks. In *International Conference on Evolutionary Computing (CEC)*, page In Press. IEEE, 2007.
110. H. G. Sandalidis, C. X. Mavromoustakis, and P. Stavroulakis. Ant based probabilistic routing with pheromone and antipheromone mechanisms. *Communication Systems*, 17:55–62, 2004.
111. H. G. Sandalidis, C. X. Mavromoustakis, and P. P. Stavroulakis. Performance measures of an ant based decentralised routing scheme for circuit switching communication networks. *Soft Computing*, 5(4):313–317, 2001.
112. R. Schoonderwoerd and O. Holland. Minimal agents for communications network routing: The social insect paradigm. *Software Agents for Future Communication Systems*, 1(1):1–2, 1999.
113. R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2):169–207, 1996.
114. T. Seeley. *The Wisdom of the Hive*. Harvard University Press, London, 1995.
115. A. U. Shankar, C. Alaettinoğlu, I. Matta, and K. Dussa-Zieger. Performance comparison of routing protocols using MaRS: Distance-vector versus link-state. In *Proceedings of ACM SIGMETRICS/PERFORMANCE*, pages 181–192, 1992.
116. C.-C Shen and C. Jaikaeo. Ad hoc multicast routing algorithm with swarm intelligence. *MONET*, 10(1-2):47–59, 2005.
117. C.-C. Shen, C. Jaikaeo, C. Srisathapornphat, Z. Huang, and S. Rajagopalan. Ad hoc networking with swarm intelligence. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *Proceedings of ANTS*, volume 3172 of *Lecture Notes in Computer Science*, pages 262–269. Springer-Verlag, 2004.
118. C.-C Shen, S. Rajagopalan, G. Borkar, and C. Jaikaeo. A flexible routing architecture for ad hoc space networks. *Computer Networks*, 46(3):389–410, 2004.
119. E. Sigel, B. Denby, and S. Le Heárat-Mascle. Application of ant colony optimization to adaptive routing in a LEO telecommunications satellite network. *Annals of Telecommunications*, 57(5–6):520–539, May-June 2002.
120. K. M. Sim and W. H. Sun. Ant colony optimization for routing and load-balancing: Survey and new directions. *IEEE Transactions on Systems, Man and Cybernetics-Part A*, 33(5):560–572, 2003.
121. K.M. Sim and W.H. Sun. Ant colony optimization for routing and load-balancing: Survey and new directions. *IEEE Transactions on Systems, Man, and Cybernetics–Part A*, 33(5):560–572, September 2003.
122. Ivan Stojmenović, editor. *Mobile Ad-Hoc Networks*. John Wiley & Sons, 2002.
123. D. Subramanian, P. Druschel, and J. Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. In *Proceedings of IJCAI*, pages 832–838. Morgan Kaufmann, 1997.
124. D. J. T. Sumpter. From bee to society: An agent-based investigation of honey bee colonies. Phd. thesis, The University of Manchester, UK, 2000.
125. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

126. S. Tadrus. *Generic multi-pheromone quality of service routing.* PhD thesis, Deaprtment of Computer Science, University of Nottingham, 2007.
127. S. Tadrus and L. Bai. A QoS network routing algorithm using multiple pheromone tables. In *Web Intelligence*, pages 132–138, 2003.
128. S. Tadrus and L. Bai. QColony: a multi-pheromone best-fit QoS routing algorithm as an alternative to shortest-path routing algorithms. *International Journal of Computational Intelligence and Applications*, 5(2):141–167, 2005.
129. G. Theraulaz and E. Bonabeau. A brief history of stigmergy. *Artificial Life,* Special Issue on Stigmergy, 5:97–116, 1999.
130. M. Thirunavukkarasu. Reinforcing reachable routes. Master thesis, Virginia Polytechnic Institue and State University, 2004.
131. R. van der Put. Routing in packet switched networks using agents. Master thesis, KBS, Delft University of Technology, Netherlands, 1998.
132. R. van der Put. Routing in the faxfactory using mobile agents. Technical report, KPN Research, 1998.
133. S. Varadarajan, N. Ramakrishnan, and M. Thirunavukkarasu. Reinforcing reachable routes. *Computer Networks*, 43(3):389–416, 2003.
134. A.V. Vasilakos and G.A. Papadimitriou. A new approach to the design of reinforcement scheme for learning automata: Stochastic Estimator Learning Algorithms. *Neurocomputing*, 7(275), 1995.
135. V. Verstraete, M. Strobbe, E. Van Breusegem, J. Coppens, M. Pickavet, and P. Demeester. AntNet: ACO routing algorithm in practice. In *Proceedings of INFORMS Telecommunications Conference*, 2006.
136. K. von Frisch. *Tanzsprache und Orientierung der Bienen.* Springer-Verlag, Heidelberg, 1965.
137. K. von Frisch. *The Dance Language and Orientation of Bees.* Harvard University Press, Cambridge, 1967.
138. S. Vutukury. *Multipath routing mechanisms for traffic engineering and quality of service in the Internet.* PhD thesis, University of California, Santa Cruz, CA, USA, March 2001.
139. Z. Wang. *Internet QoS: Architectures and Mechanisms for Quality of Service.* Morgan Kaufmann, 2001.
140. H. F. Wedde and M. Farooq et.al. BeeAdHoc–An Energy-Aware Scheduling and Routing Framework. Technical report-pg439, LSIII, School of Computer Science, University of Dortmund, 2004.
141. H. F. Wedde and M. Farooq. Beehive: New ideas for developing routing algorithms inspired by honey bee behavior. In Albert Y. Zomaya Stephan Olariu, editor, *Handbook of Bioinspired Algorithms and Applications*, chapter 21, pages 321–339. Chapman & Hall/CRC Computer and Information Science, 2005.
142. H. F. Wedde and M. Farooq. BeeHive: Routing algorithms inspired by honey bee behavior. *Künstliche Intelligenz,* Special Issue on Swarm Intelligence, 4:18–24, November 2005.
143. H. F. Wedde and M. Farooq. A performance evaluation framework for nature inspired routing algorithms. In *Applications of Evolutionary Computing*, volume 3449 of *LNCS*, pages 136–146. Springer, 2005.
144. H. F. Wedde and M. Farooq. The wisdome of the hive applied to mobile ad-hoc networks. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 341–348, 2005.

145. H. F. Wedde and M. Farooq. A comprehensive review of nature inspired routing algorithms for fixed telecommunication networks. *Journal of System Architecture*, 52(8-9):461–484, 2006.
146. H. F. Wedde, M. Farooq, T. Pannenbaecker, B. Vogel, C. Mueller, J. Meth, and R. Jeruschkat. BeeAdHoc: an energy efficient routing algorithm for mobile ad-hoc networks inspired by bee behavior. In *Proceedings of GECCO*, pages 153–161, 2005.
147. H. F. Wedde, M. Farooq, C. Timm, J. Fischer, M. Kowalski, M. Langhans, N. Range, C. Schletter, R. Tarak, M. Tchatcheu, F. Volmering, S. Werner, and K. Wang. BeeAdHoc–An Efficient, Secure, Scalable Routing Framework for Mobile AdHoc Networks. Technical report-pg460, LSIII, School of Computer Science, University of Dortmund, 2005.
148. H. F. Wedde, M. Farooq, and Y. Zhang. BeeHive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. In *Ant Colony Optimization and Swarm Intelligence*, volume 3172 of *Lecture Notes in Computer Science*, pages 83–94. Springer Verlag, Sept 2004.
149. H. F. Wedde, C. Timm, and M. Farooq. BeeHiveAIS: A simple, efficient, scalable and secure routing framework inspired by artificial immune systems. In *Proceedings of the PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 623–632. Springer Verlag, September 2006.
150. H. F. Wedde, C. Timm, and M. Farooq. BeeHiveGuard: A step towards secure nature inspired routing algorithms. In *Applications of Evolutionary Computing*, volume 3907 of *Lecture Notes in Computer Science*, pages 243–254. Springer Verlag, April 2006.
151. T. White. Swarm intelligence and problem solving in telecommunications. *Canadian Artificial Intelligence Magazine*, (41):14–16, 1997.
152. T. White, B. Pagurek, and F. Oppacher. ASGA: Improving the ant system by integration with genetic algorithms. In *Proceedings of the Third Annual Conference on Genetic Programming*, pages 610–617, 1998.
153. T. White, B. Pagurek, and F. Oppacher. Connection management using adaptive mobile agents. In H.R. Arabnia, editor, *Proceedings of the PDPTA*, pages 802–809. CSREA Press, 1998.
154. T. White, B. Pagurek, and F. Oppacher. Application oriented routing with biologically-inspired agents. In *Proceedings of GECCO*, pages 610–617, 1999.
155. Y. Yang, A. N. Zincir-Heywood, M. I. Heywood, and S. Srinivas. Agent-based routing algorithms on a LAN. In *IEEE Canadian Conference on Electrical and Computer Engineering*, 1442–1447 2002.
156. S. Zahid, M. Shehzad, S. Usman Ali, and M. Farooq. A comprehensive formal framework for analyzing the behavior of nature inspired routing protocols. In *Proceedings of CEC*. In Press, IEEE Press, 2007.
157. L. Zhang, S. Deering, and D. Estrin. RSVP: A new resource ReSerVation protocol. *IEEE Networks*, 7(5):8–18, September 1993.
158. R. Zhang and M. Bartell. *BGP design and implementation*. CISCO Press, 2003.
159. Z. Zhang, C. Sanchez, B. Salkewicz, and E. Crawley. Quality of service extensions to OSPF. Internet Draft draft-zhang-qos-ospf-00, Internet Engineering Task Force (IEFT), June 1996.